UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de grado

Diseño y modelado de elementos 3D para la gamificación en educación especial

David Redondo Pérez

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Diseño y modelado de elementos 3D para la gamificación en educación especial

Autor: David Redondo Pérez
Tutor: Javier Gómez Escribano
Ponente: Germán Montoro Manrique

julio 2020

David Redondo Pérez
Diseño y modelado de elementos 3D para la gamificación en educación especial

David Redondo Pérez
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia y amigos
por todo el apoyo recibido

El verdadero progreso es el que pone la tecnología al alcance de todos.

Henry Ford

Agradecimientos

En primer lugar, me gustaría agradecer a Javier Gómez Escribano, mi tutor, toda la ayuda que me ha aportado durante el transcurso de la carrera. Gracias a la paciencia que ha tenido conmigo en los momentos más duros y al gran trabajo realizado resolviéndome las dudas, he podido finalizar este proyecto.

Gracias en especial a mi familia, quienes han hecho todo lo que ha estado en su mano para que yo pudiera estudiar. Nunca me han dejado rendirme y me han animado a seguir adelante por muy duro que fuera la situación. Hemos compartido muchos buenos momentos y otros no tanto. Es por ello, que el resultado de mis logros académicos es tanto mío como vuestro.

Gracias a todos mis compañeros de carrera, de los cuales muchos de ellos han terminado convirtiéndose en amigos con el traspaso del tiempo. Hemos pasado por muchos momentos de risas, lloros, aprobados y suspensos, pero ante todo siempre nos hemos apoyado entre nosotros tanto fuera como dentro de clase.

Gracias a todos mis amigos de toda la vida, que, aunque no les haya podido ir a ver todo lo que me hubiera gustado durante la carrera siempre han estado cuando se les ha necesitado.

Por último, pero no menos importante, gracias a todas aquellas personas que durante mis años de estudio me han puesto impedimentos y piedras en el camino con la intención de desmotivarme. Me ha servido de motivación para demostrar que con esfuerzo y sacrificio las cosas se pueden conseguir.

Resumen

La Realidad Aumentada, durante los últimos años, ha ido ganando importancia gracias a sus múltiples utilidades en diversos sectores. Es por ello por lo que, con la proliferación del uso de dispositivos móviles, hayan surgido diversas plataformas que hacen uso de esta tecnología en sus aplicaciones. Una de ellas, ha sido la creada por Google (ARCore) junto con su plugin Sceneform, la cual su principal intención es llegar a la mayor cantidad posible de usuarios facilitando y simplificando a los desarrolladores las tareas de trabajo con modelados 3D, ya que en muchos casos pueden resultar complejas. Estas herramientas se han utilizado en diversos ámbitos de la sociedad como pueden ser la sanidad, el ocio, la industria o la educación.

El Trastorno del Espectro Autista (TEA), es un trastorno de origen neurobiológico que afecta al sistema nervioso y funcionamiento cerebral. Es por ello por lo que las personas con TEA sufren de dificultades a la hora de aprender y relacionarse con los demás.

El objetivo principal de este trabajo es analizar la situación actual de la Realidad Aumentada para dispositivos móviles cuyo sistema operativo sea Android. Para ello se ha desarrollado una aplicación de muestra reutilizable haciendo uso de las principales funcionalidades que ofrecen estas herramientas. Dicho software, se plantea para poder ser integrado en un futuro en aplicaciones educativas para niños con TEA. Esta tecnología servirá como foco de motivación premiando al alumno con elementos visuales e interactivos durante la corrección de los ejercicios que se hayan resuelto satisfactoriamente.

Palabras clave

Realidad Aumentada, ARCore, Sceneform, TEA, Dispositivos móviles

Abstract

Augmented Reality, during the last years, has been gaining importance due to its multiple utilities in several fields. For this reason, with the proliferation of the use of mobile devices, various platforms have emerged which use this technology in their applications. One of them has been developed by Google (ARCore) together with its Sceneform plugin, whose main intention is to reach the largest possible number of users, making it easier and simpler for developers to work with 3D models, since in many cases they can be complex. These tools have been used in various areas of society such as health, leisure, industry or education.

Autism Spectrum Disorder (ASD) is a neurobiological disorder that affects the nervous system and brain function. For this reason, people with ASD suffer from difficulties in learning and relating with others.

The main objective of this project is to analyze the current situation of Augmented Reality for mobile devices whose operating system is Android. For this purpose, a reusable sample application has been developed using the main functionalities offered by these tools. This software is designed to be integrated into future educational applications for children with ASD. This technology will serve as a focus of motivation by rewarding the student with visual and interactive elements as part of the feedback of the exercises that have been solved correctly.

Keywords

Augmented Reality, ARCore, Sceneform, TEA, Mobile devices

Índice

Listas

Lista de códigos

Lista de figuras

Lista de tablas

1

Introducción

1.1. Motivación

Durante las últimas décadas la investigación y el desarrollo de aplicaciones que hacen uso de la Realidad Aumentada se han visto potenciadas debido a la alta aceptación por parte de la población. Dicha aceptación ha sido posible al tratarse de una tecnología altamente llamativa para los usuarios, capaz de combinar información y elementos digitales (imágenes, objetos 3D, etc.) en un entorno real de forma muy visual e interactiva. Por dichos motivos la Realidad Aumentada ha emergido con fuerza en diferentes sectores de la sociedad como pueden ser la industria, el ocio, la medicina o la educación.

En el ámbito educativo ha supuesto un gran avance en el proceso de aprendizaje debido a que la utilización de esta tecnología es un componente de motivación esencial, personalizable e interactivo, facilitando la transmisión de conocimientos de una forma mucho más rápida y fácil de comprender. Dichas características provocan mayor iniciativa por parte del estudiante a la hora de aprender. Un añadido es que los dispositivos en los cuales se suelen desarrollar este tipo de aplicaciones suelen ser dispositivos móviles (iOS y Android) a los cuales tienen acceso la mayor parte de la población, esto hace que la Realidad Aumentada sea una tecnología con futuro.

En el caso de la educación especial para niños con Trastornos del Espectro Autista (TEA), se pueden conseguir grandes mejoras haciendo uso de la Realidad Aumentada ya que permite al equipo docente realizar técnicas pedagógicas innovadoras creando contenido personalizado, visual e interactivo. Los niños con TEA en su gran mayoría sufren dificultades del desarrollo cognitivo por lo que el uso de esta tecnología puede resultar muy beneficiosa, mejorando la asimilación de conocimientos por parte del niño, así como conseguir una mayor motivación por el aprendizaje. Para conseguir dichas mejoras y no producir rechazo por parte de los usuarios es necesario tener unas bases sólidas en el desarrollo de aplicaciones de Realidad Aumentada tanto en su aspecto de análisis, diseño e implementación, como en el modelado de objetos 3D y su posterior integración en la herramienta.

1.2. Objetivos

El principal objetivo de este proyecto es el estudio de la Realidad Aumentada en la actualidad desarrollando un caso de muestra para su correcta comprensión dando la posibilidad de reutilizarse en aplicaciones futuras que requieran el uso de esta tecnología.

La finalidad principal del software de prueba es el aprendizaje del flujo de trabajo necesario para el correcto desarrollo de una aplicación de Realidad Aumentada partiendo desde el software utilizado para dicho desarrollo, pasando por su fase de implementación junto con la fase de integración de los diferentes modelos 3D y sus correspondientes formatos, y por último, su fase final de pruebas.

El desarrollo de esta aplicación está pensada para servir de modulo auxiliar en una futura integración de esta tecnología en la aplicación de matemáticas [1] utilizada actualmente por los alumnos con TEA. Esta sección de la aplicación servirá de motivación adicional para los usuarios al evaluar un ejercicio realizado previamente de forma correcta, recibiendo como recompensa personajes 3D que sean agradables y animen al alumno que termina el ejercicio satisfactoriamente.

1.3. Estructura del Documento

La estructura del documento se encuentra organizada en varios capítulos. En el **Capítulo 2** se definirá la Realidad Aumentada y se analizarán las diferentes herramientas existentes en el mercado diferenciando sus características principales, para finalmente elegir la que mejor se adapte al proyecto. En el **Capítulo 3** se desarrollará un aplicación de muestra haciendo uso esta tecnología. Se explicará la funcionalidad de la aplicación, la creación del proyecto y los conceptos básicos de modelado 3D necesarios para una correcta integración de avatares en la aplicación. Por último, se realizará la implementación de dicho prototipo. En el **Capítulo 4** se mostrarán todas las pruebas realizadas durante el desarrollo del proyecto y para finalizar, en el **Capítulo 5** se expondrán las conclusiones alcanzadas tras el trabajo realizado.

2

Análisis de la Realidad Aumentada en dispositivos móviles

En este capítulo se explicará el concepto de Realidad Aumentada y a continuación, se analizará la situación actual en dispositivos móviles realizando una comparación entre las distintas variantes existentes en el mercado de esta tecnología para dispositivos Android.

2.1. La Realidad Aumentada

La primera vez que se utilizó este termino fue en 1990 por Boeing Tom Caudel. De acuerdo con el artículo Recent advances in augmented reality [2] un sistema de Realidad Aumentada es aquel que complementa el mundo real con objetos virtuales (generados por ordenador) que parecen coexistir en el mismo espacio que el mundo real.

Sin embargo, los primeros en hablar sobre esta tecnología fueron Milgram y Kishino [3] los cuales describieron una taxonomía para enlazar un entorno completamente real con uno virtual. Por ello, todos los posibles estados situados entre ambos extremos se corresponderían con una Realidad Mixta ya que poseen elementos de ambos mundos. Como se puede observar en la figura 2.1, la Realidad Aumentada se sitúa más cercana a un entorno real ya que añade elementos 3D a una situación física.

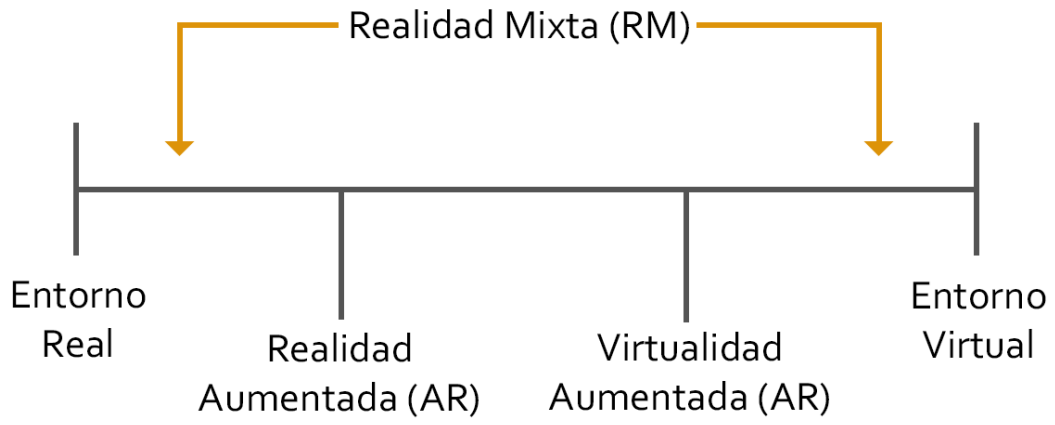


Figura 2.1: Esquema la Realidad Mixta

Como se muestra en el estudio de Ronald T. Azuma [4], todo sistema de Realidad Aumentada debe reunir 3 principales características: Ser capaz de combinar elementos virtuales con reales, ser interactivo en tiempo real y, por último, rastrear y registrar en 3 dimensiones.

En dicho artículo se muestra la gran variedad de sectores en los que el uso de esta tecnología puede resultar muy beneficiosa como puede ser la medicina, la industria, el ocio, o la educación. En el ámbito educativo ha permitido la investigación de nuevas metodologías didácticas que posibilitan nuevas formas de enseñanza que complementen a las actuales.

En el caso de los niños con TEA, el uso de esta tecnología puede servir de gran ayuda, ya que, al sufrir de dificultades en el aprendizaje, la Realidad Aumentada puede afianzar y facilitar el proceso de entendimiento y comprensión de conocimientos. Un ejemplo es el estudio realizado por J. C. Torrado, J. Gomez, and L. Jaccheri [1] en el cual se hace uso de la Realidad Aumentada para ayudar a niños con TEA a ser capaces de autoevaluarse para mejorar su autonomía y autodeterminación.

2.2. Características Técnicas

Para la correcta compresión de las diferencias entre plataformas existentes en el mercado de Realidad Aumentada, se presenta una explicación previa de las diversas funcionalidades o características técnicas que suelen poseer estas herramientas ya sean en mayor o menor medida.

Rastreo de Imágenes

También llamado **Image Tracking**, es la capacidad de reconocer una imagen 2D en un entorno real haciendo uso de la cámara del dispositivo mientras te mueves. Antiguamente se necesitaban imágenes con determinados patrones como los códigos QR, sin embargo, en la actualidad, se puede utilizar casi cualquier imagen que contenga suficiente contraste y detalle. Una funcionalidad añadida es el Rastreo Extendido, la cual permite al dispositivo mantener los modelados 3D en un plano virtual evitando que se pierdan dichos objetos cuando la aplicación deja de rastrear la imagen.

Rastreo de Planos

También llamado **Plane Tracking**, es la capacidad de reconocer superficies planas tanto verticales como horizontales. Un ejemplo muy sencillo es la detección del suelo, paredes, mesas, etc. De esta forma podemos representar modelos 3D sin dar la sensación de que se encuentran flotando en el aire.

Estimación de Iluminación

También llamado **Lighting Estimation**, es la capacidad de saber cuál es la luminosidad de la escena para poder aplicarle posteriormente la misma cantidad de luz al modelo 3D renderizado. De esta manera se consigue una mayor inmersión ya que la escena logra

un mayor realismo. Sin embargo, esta característica tiene sus limitaciones ya que en la mayoría de las plataformas se conoce la luminosidad pero no su dirección.

Mundo Compartido

También llamado **Shared Worlds**, es la capacidad de compartir una comprensión del entorno entre varios dispositivos permitiendo experiencias grupales entre varios usuarios. Es una característica muy reciente que se encuentra en desarrollo.

Rastreo de Caras

También llamado **Face Tracking**, es la capacidad de reconocer y seguir rostros humanos en movimiento. Esto permite añadir elementos visuales para modificar el aspecto físico del usuario.

Rastreo de Cuerpos

También llamado **Body Tracking**, es la capacidad de rastrear el cuerpo de una persona, ya sean sus brazos, piernas, manos ,etc. Esta tecnología existe desde hace tiempo en dispositivos como Xbox Kinect en el cual se permitía jugar moviendo el cuerpo. Sin embargo, esta característica cobra un mayor sentido en aquellos aparatos que no sujetamos con las manos por lo que en este caso no se hará uso de esta propiedad.

Rastreo de Objetos 3D

También llamado **3D Object Tracking**, es la capacidad de reconocer objetos 3D como mesas, vasos, lamparas, etc. Esta tecnología supone una mayor dificultad en el desarrollo, puesto que, para detectar el objeto en el entorno real, debe haber sido precargado antes.

Oclusión

También llamado **Occlusion**, es la capacidad de relacionar los objetos 3D de forma realista con la escena. En la mayoría de las situaciones se muestra el modelo por encima de la vista real, sin embargo, con esta característica si un objeto de la pantalla se mueve detrás de un elemento real, el cual lo debería tapar, este se ocultará creando la ilusión óptica deseada.

Base de Datos en la Nube

También llamado **Cloud Database**, es la capacidad de actualizar las imágenes y los objetos 3D de la aplicación de forma online. Aunque no todas las plataformas del mercado lo permiten, desarrollar esta funcionalidad de forma personalizada no es demasiado complicado.

Geoetiquetado

También llamado **Geo-tagging**, es la capacidad de introducir contenido de Realidad Aumentada en lugares reales del mapa. Este tipo de tecnología se encuentra actualmente en unas fases muy tempranas de su desarrollo. Un ejemplo de este tipo de tecnología es el popular juego de móviles PokemonGo.

2.3. Kits de desarrollo AR

Una vez comprendidas las diferentes características que suelen tener este tipo de tecnología, a continuación, se presenta un estudio de los principales kits de desarrollo que existen actualmente en el mercado, centrándose sobre todo en aquellas compatibles con Android, puesto que la intención es llegar al máximo número de usuarios.

2.3.1. Easy AR

Easy AR [5] es un kit de desarrollo de Realidad Aumentada compatible con sistemas operativos tanto Android como iOS. Destaca por ser una de las plataformas más utilizada en el mercado junto a ARCore, la cual incluye principalmente rastreo de imágenes, rastreo de superficies y su característica más diferenciadora del resto de herramientas, **rastreo de objetos 3D**.

Easy AR, al igual que la mayoría de las plataformas, trabaja bajo la librería gráfica OpenGL ES, en concreto GLES2. OpenGL ES (GLES) es una rama de la interfaz OpenGL [6] centrada en la renderización de gráficos informáticos en 2D y 3D para videojuegos en sistemas empotrados como dispositivos móviles. Para el desarrollo de cualquier software que haga uso de Easy AR es necesario un amplio conocimiento de esta librería gráfica ya que su utilización puede llegar a ser complicada. Además, se deben cumplir los siguientes requisitos mínimos tanto en los dispositivos en los que se despliega la aplicación como en el entorno de desarrollo.

- JDK 1.8
- Android NDK r19c
- Android 4.0
- Android SDK Build Tools 28.0.3

Soporta los lenguajes de programacion C, C++, Java, Kotlin, Swift y Objective-C. Por otro lado, Unity [1], el famoso motor de videojuegos soporta esta herramienta para sus aplicaciones de Realidad Aumentada.

Existen varias licencias del producto dependiendo de las necesidades del desarrollador partiendo desde una gratuita que incluye las funcionalidades básicas hasta una licencia empresarial con la funcionalidad completa.

2.3.2. Spark AR

Spark AR [7] es la plataforma de Realidad Aumentada creada por Facebook y aunque posee rastreo de imágenes y rastreo de planos, es realmente conocida por su funcionalidad de **rastreo de caras**.

[1] `https://unity.com/es` (08/07/2020)

Instagram hace uso de esta herramienta con los populares filtros personalizados en los cuales se pueden poner elementos visuales para deformar la apariencia física de la persona que lo usa.

Para la implementación de aplicaciones que utilizan Spark AR hay que hacer uso de Spark AR Studio, la cual es una herramienta de desarrollo que facilita en gran medida la creación de este tipo de software a los programadores. Esto es debido a que su programación se realiza de forma muy visual, arrastrando y utilizando contenedores que contienen internamente la funcionalidad deseada (animaciones, shaders, funciones lógicas, etc.) En este caso, el lenguaje soportado es Javascript. Gracias a estas características, Spark Ar se ha convertido en una plataforma a tener en cuenta para proyectos de pequeño tamaño debido a su rápido y fácil desarrollo. La licencia que ofrece Spark AR para la creación de contenido es completamente gratuita.

2.3.3. Vuforia

Vuforia [8] es una herramienta de Realidad Aumentada centrada principalmente en el reconocimiento de imágenes, objetos y entornos. Sin embargo la característica más diferenciadora de esta plataforma sobre las otras es **Vuforia Fusion**, la cual combina un conjunto de tecnologías de RA para proporcionar la mejor experiencia posible en una amplia gama de dispositivos.

Veamos esta herramienta con el siguiente esquema.

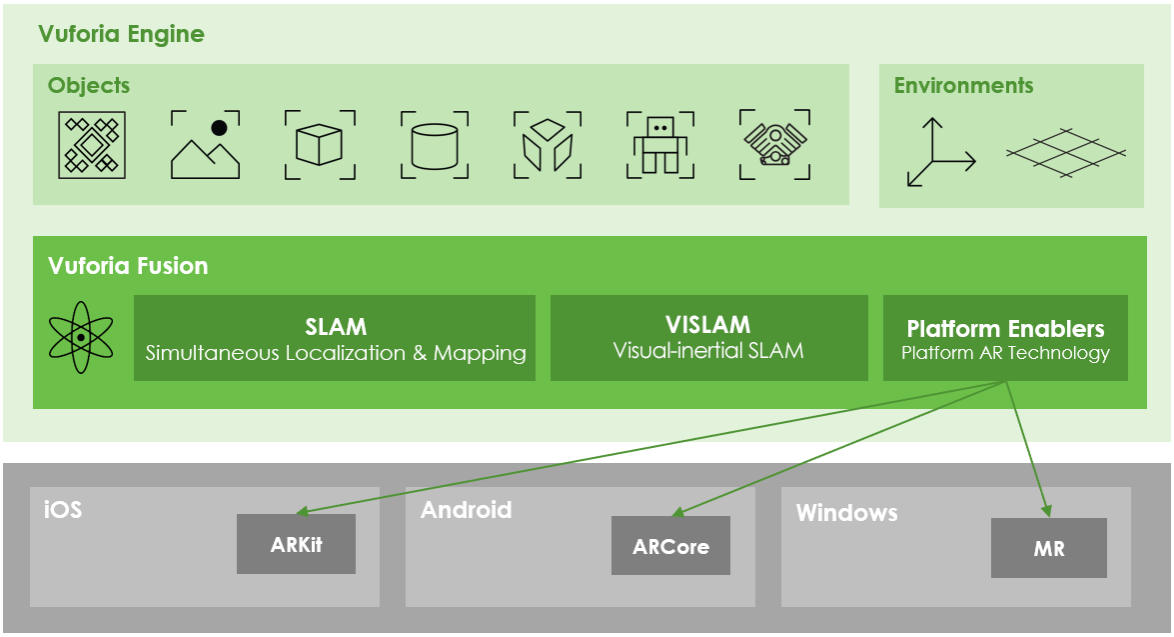


Figura 2.2: Esquema de Vuforia Fusion

Una aplicación desarrollada con Vuforia aprovechará los puntos fuertes de las otras librerías compatibles con el dispositivo. Si se ejecuta una aplicación sobre un dispositivo Android con ARCore ins-

talado, hará uso de sus funcionalidades. Por contra, si el dispositivo no es compatible, usará Vuforia VISLAM si se poseen los sensores necesarios.

Los requisitos mínimos que se deben cumplir son los siguientes:

- JDK 1.8
- Android NDK r20+
- Android 5.1.1
- Android Studio 3.4
- Android SDK Build Tools 28.0.3

Los principales lenguajes soportados por Vuforia son C++ y Java. En el apartado de gráficos, al igual que Easy Ar y AR Core, soporta GLES2 y GLES3 además de Vulkan. Por último,a diferencia del resto de plataformas analizadas, Vuforia no ofrece ninguna licencia gratuita.

2.3.4. ARCore

ARCore [9] es la herramienta creada por Google en 2017 para el mercado de Realidad Aumentada. Al igual que el resto de las plataformas, entre las características más importantes se encuentran: rastreo de imágenes, rastreo de planos, animaciones, seguimiento de caras, etc. ARCore además, es capaz de calcular la iluminación de la escena dando un mayor realismo a los objetos 3D representados. Otra característica innovadora de ARCore ha sido la introducción de experiencias compartidas entre usuarios ampliando las posibilidades de los desarrolladores.

Para el posicionamiento de modelos 3D se hace uso de nodos situados en el plano mediante puntos de anclaje de forma jerárquica. De esta forma se permite a los desarrolladores crear composiciones de varios objetos en una misma escena. Ejemplo: Si creamos un cubo encima de una mesa, el nodo compuesto por la figura tendrá como punto de anclaje la posición de la mesa en la que el usuario haya pulsado. Por otro lado, los objetos e imágenes utilizados en la aplicación se pueden almacenar tanto de forma interna, como en la nube.

La principal intención de ARCore es llegar al máximo número de dispositivos y desarrolladores facilitando varios lenguajes de programación como: Java, C++, Unity, Unreal, iOS. Sin embargo, el ecosistema de Android se encuentra muy fragmentado por lo que el funcionamiento de esta tecnología puede resultar un desafío ya que los dispositivos poseen hardware y softwares muy diferentes.

Los requisitos mínimos para el correcto funcionamiento son:

- JDK 1.8
- Android API 24
- Android Studio 3.1

Al igual que Easy AR, la librería gráfica sobre la que trabaja ARCore es OpenGL (GLES2 y GLES3). Como se ha comentado anteriormente el uso de esta herramienta puede llegar a ser muy densa y es necesario un alto conocimiento de la librería. Por ello, para facilitar todo el trabajo de renderizado de escenas 3D existe el siguiente plugin para Android Studio llamado Scenform. Este plugin evita al desarrollador trabajar directamente sobre OpenGL haciendo que la implementación de aplicaciones de Realidad Aumentada resulte mucho más sencilla.

Sceneform

SceneForm [10] aporta una API gráfica de alto nivel para el desarrollo de aplicaciones de Realidad Aumentada en ARCore desde el cual se puede importar, ver y crear elementos 3D de una forma muy sencilla y dinámica. Actualmente los formatos de elementos 3D que permite son: .fbx, .obj y .gltf. El plugin una vez se ha importado el objeto genera un modelo con formato .sfb para su posterior integración en la aplicación.

2.4. Formatos de Objetos 3D

En esta sección se muestran algunos de los formatos de objetos 3D más utilizados en aplicaciones de Realidad Aumentada. Todos los que se presentan a continuación son soportados por el plugin Sceneform.

OBJ

La extensión OBJ es conocido como Wavefront 3D Object y está formado por caracteres ASCII por lo que se tratan de archivos legibles. OBJ permite el uso de materiales y texturas propias para su representación, sin embargo no soporta animaciones al tratarse de modelos estáticos. A continuación, se muestran las principales ventajas y desventajas de este formato.

- Ventajas
  - Formato abierto
  - Rápido y eficiente
  - Facil implementación
  - Gran Compatibilidad
  - Pensado para objetos estáticos
- Desventajas
  - No soporta animaciones
  - No soporta sistema de iluminación
  - No soporta cámaras

FBX

FBX es un formato de datos 3D y aunque su uso es gratuito, es propiedad de Autodesk. El objetivo de FBX es preservar la calidad del archivo original pudiendo almacenar datos de posición, mapeado de UVs y normales, además de posibilitar su utilización en todo tipo de aplicaciones 3D. Por último, los datos que puede almacenar este formato pueden ser tanto ASCII como binarios. A continuación, se muestran las principales ventajas y desventajas.

- Ventajas
  - Rápido y eficiente (formato binario)
  - Gran almacenamiento de informacion (texturas, uvs, animaciones, etc.)
  - Modelados muy precisos
  - Muy buen SDK (Software development kit)
  - Buena Compatibilidad
  - Amplio soporte de características
- Desventajas
  - Formato cerrado
  - Características anticuadas o en desuso.
  - SDK demasiado pesado para moviles o motores de videojuegos.
  - Modelo de iluminación antiguo.
  - Modelo de materiales antiguo.

GLTF

GLTF es un formato de modelos 3D pensado principalmente para ganar en rendimiento minimizando tanto el tamaño de los objetos 3D como el tiempo de desempaquetado y utilización durante la ejecución. Se trata de un formato de publicación extensible que agiliza los flujos de trabajo llegando a ser un estándar en la industria. A continuación, se muestran las principales ventajas y desventajas.

- Ventajas
  - Rápido y eficiente (formato binario)
  - Gran almacenamiento de informacion (texturas, uvs, animaciones, etc.)
  - Facil de leer y escribir
  - Lecturas directas para los motores de videojuegos.
  - Orientación de un grupo de normas
  - Estandar para aplicaciones de Realidad Aumentada.
- Desventajas
  - Modelos 3D no editables
  - No hay redes de sombreado (mal formato para materiales que se desean editar)
  - Fragmentación debido las extensiones específicas.

Actualmente Sceneform no permite las animaciones de este formato. Sin embargo, sí que se pueden utilizar los objetos sin necesidad de transformarlos al formato .sfb.

Por tanto, se puede concluir que dependiendo de la situación puede ser conveniente utilizar un formato u otro. Si lo que se quiere es modelar un objeto simple sin animaciones puede ser buena idea utilizar obj. Si, por el contrario, lo que se pretende es mantener el máximo detalle con o sin animaciones sería conveniente utilizar fbx. En el caso de que se busque el máximo rendimiento de la aplicación y se sepa que no se va a modificar el modelo a posteriori, el mejor formato es gltf.

2.5. Conclusiones

Una vez analizadas las diferentes variantes existentes en el mercado se puede concluir que dependiendo de los objetivos puede resultar más interesante utilizar una opción u otra ya que todas comparten una serie de características comunes. Para la elección se debe tener en cuenta varios factores como: a que público va dirigido, cual es el conocimiento de la librería gráfica por parte del desarrollador, cual es la característica con mayor potencial de la herramienta, coste de licencias, dispositivos compatibles, etc. En la tabla 2.1 se puede observar lo explicado anteriormente de forma resumida.

	ARCore	**Easy AR**	**Vuforia**	**Spark Ar**
Rastreo de Imagenes	SI	SI	SI	SI
Rastreo de Planos	SI	SI	SI	SI
Estimación de Iluminacón	SI			
Mundos Compartidos	SI			
Rastreo de Caras				SI
Rastreo de Cuerpos				Parcial
Sistema Operativo	Android / iOS	Andrid / iOS	Android / iOS	Facebook / Instagram
Licencia	Gratuita	Gratuita / Pago	Pago	Gratuita

Tabla 2.1: Esquema de funcionalidades principales de cada kit de desarrollo

Por ello, para este proyecto se ha decidido utilizar la plataforma ARCore puesto que es compatible con la mayor parte de dispositivos Android existentes en el mercado. Además, posee todas las herramientas que se necesitan como el rastreo de imagenes, rastreo de planos, iluminación, etc. Por último, gracias al plugin Sceneform todo el apartado de importacion y visualización de objetos 3D resultará menos complejo sin ser necesario conocer el funcionamiento de OpenGL.

3

Desarrollo de una aplicación de Realidad Aumentada en Android

En este capítulo se explicará todo el proceso que es necesario llevar a cabo para desarrollar la aplicación (**RABase**) de Realidad Aumentada, desde la creación del proyecto, fase de modelado y por último, su implementación. Dicha aplicación además, servirá de modelo para aquellas aplicaciones educativas especializadas en niños con TEA que deseen hacer uso de esta tecnología puesto que, únicamente será necesario importar las clases con sus dependencias para el correcto funcionamiento.

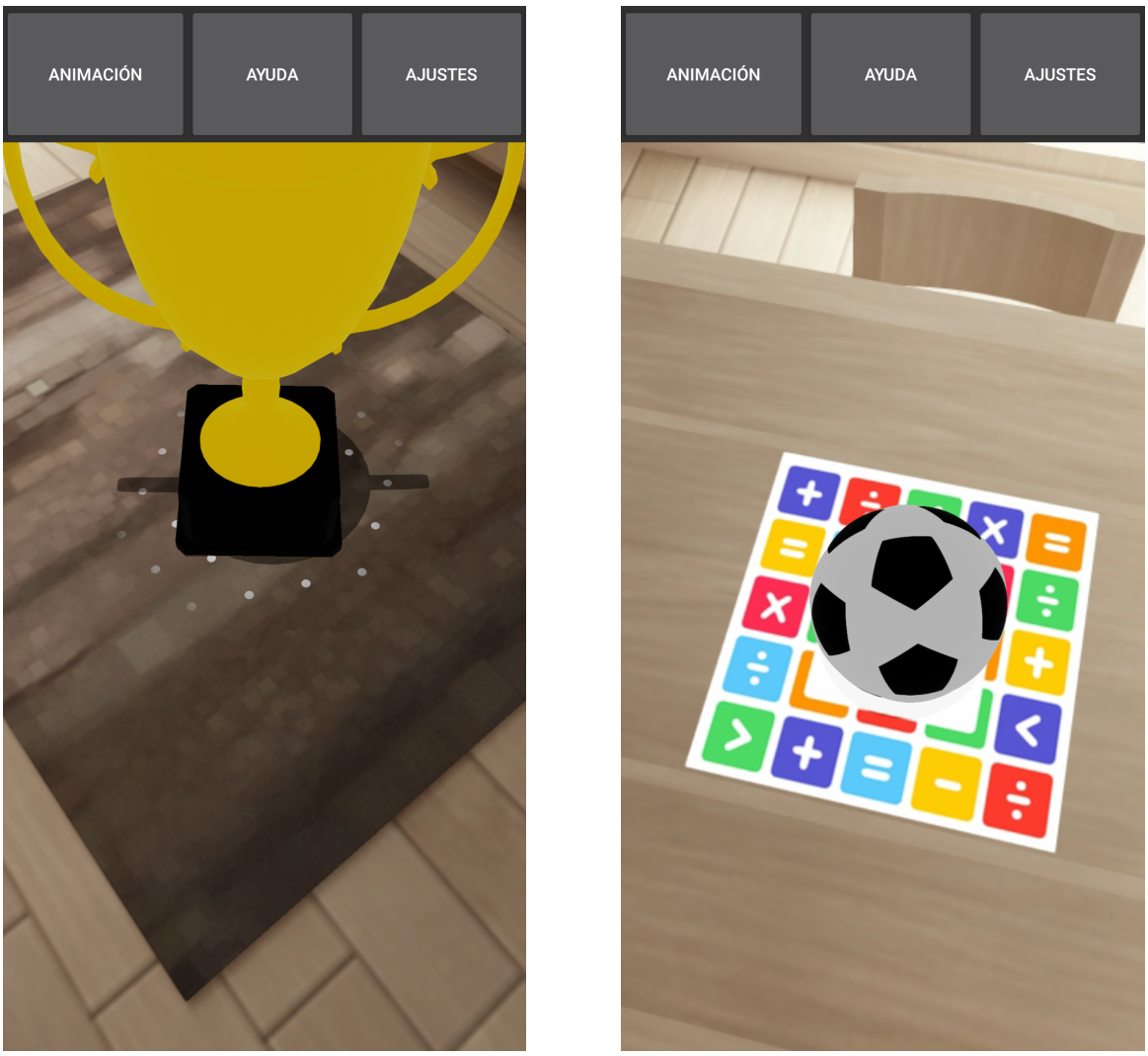
3.1. Descripción y Funcionalidad

La aplicación RABase fue planteada desde un primer momento como un prototipo aplicable a proyectos para niños con TEA que deseen trabajar con Realidad Aumentada, introduciendo avatares y objetos 3D en sus aplicaciones a modo de recompensa para los estudiantes.

RABase se encuentra desarrollada en Android haciendo uso de ARCore y Sceneform. Está formada por una actividad principal, en la que se observa un menú en la parte superior de la pantalla con diferentes opciones: Ajustes, Ayuda y Animación. Por otro lado, en el espacio sobrante se muestra el fragmento de Realidad Aumentada en el cual se hará uso de la cámara para el reconocimiento del entorno.

Para la representación de objetos 3D se encuentran disponibles dos métodos de renderizado. El primero de ellos es mediante **rastreo de planos** en el cual el usuario buscará terrenos planos ya sean verticales u horizontales rastreables y podrá crear objetos en aquellos puntos de la superficie que desee. Esta característica se puede observar en la figura 3.1(a).

Por otro lado se encuentra el **rastreo de imágenes**. El usuario utilizará la cámara en busca de imágenes precargadas previamente. De esta forma cuando la aplicación detecte la imagen deseada, renderizará el objeto 3D en el centro de la imagen. Esta característica se puede observar en la figura 3.1(b). Como imagen de detección se ha hecho uso de uno de los dibujos mostrados en las fichas de ejercicios de la aplicación de matemáticas comentada anteriormente. Esto se debe a la intención de poder añadir la Realidad Aumentada en dicha aplicación.



(a) Detección de plano

(b) Detección de imagen

Figura 3.1: En las siguientes subfiguras se muestran los diferentes métodos de representación de un avatar. La 3.1(a) es un ejemplo de renderizado de un objeto pulsando en un punto cualquiera del plano detectado. La 3.1(b) es una muestra de renderizado de un objeto mediante la detección de una imagen establecida previamente.

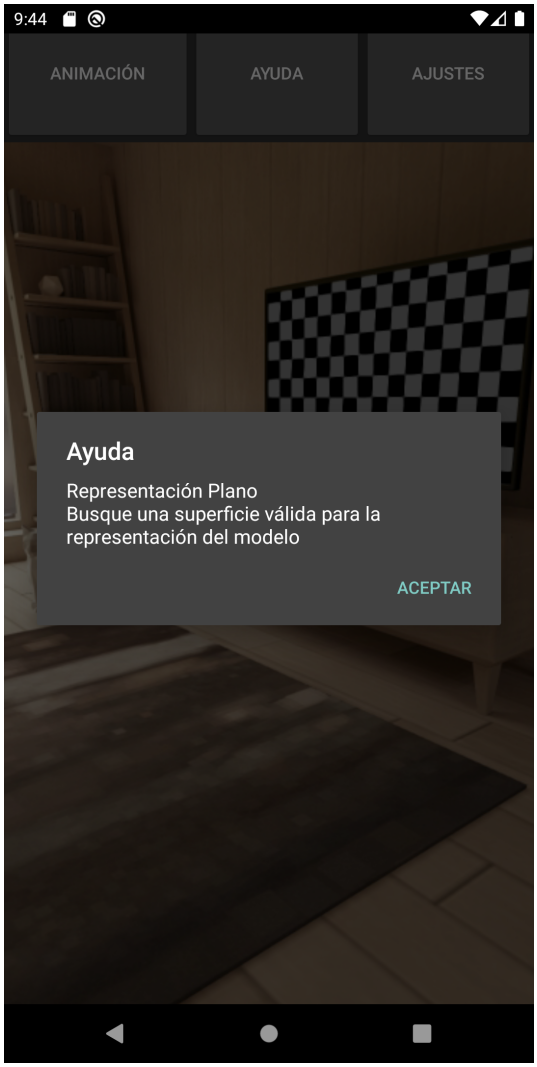
Cuando el usuario pulsa el **botón de ayuda** se le mostrará un mensaje informativo sobre los pasos que debe realizar para la representación de los elementos visuales deseados. Esta característica se puede observar en la figura 3.2(a)

Cuando el usuario selecciona en la barra superior del menu la **opción de ajustes** se mostrará una pantalla con todas las configuraciones disponibles de la aplicación. Dichos ajustes vienen diferenciados en dos bloques principales. En el primero se encuentran las configuraciones tanto del método de representación, como de selección del diseño que se desea representar. Hay un total de 5 diseños en formatos diferentes (obj, fbx, gltf). De esta manera podemos observar las principales características de cada uno de ellos, ya sean texturas, animaciones, complejidad, etc. En el segundo bloque se encuentran las configuraciones más técnicas de la aplicación como pueden ser el autoenfoque de la cámara, la estimación de luminosidad, el dibujado del plano detectado y el dibujado de sombras. Finalmente se muestra la imagen que se utilizará en el rastreo de imágenes junto con información adicional de la aplicación. Esta pantalla se muestra en la figura 3.2(b)

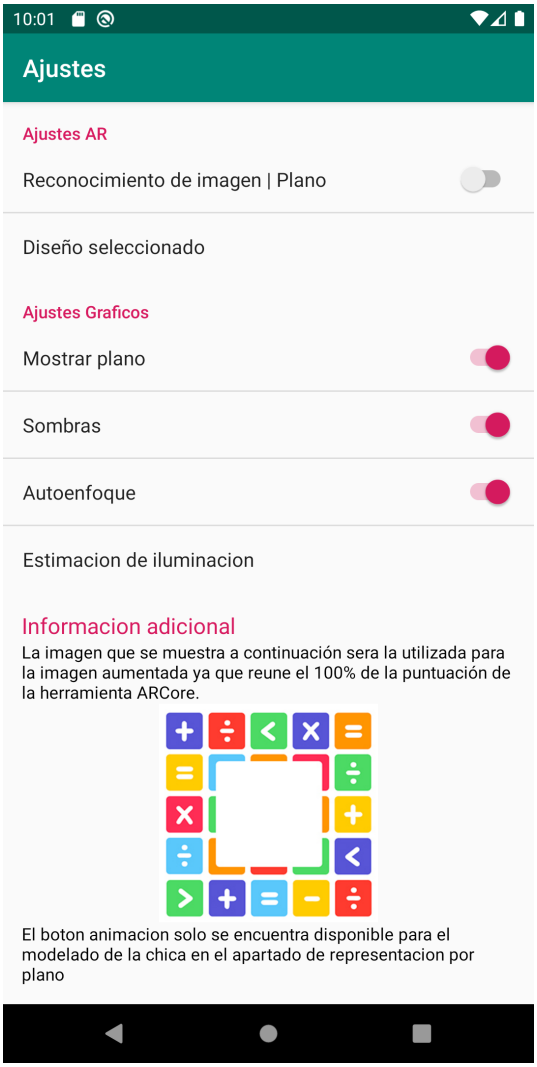
3.2. Creación del Proyecto

Para el desarrollo de esta aplicación se ha decidido utilizar Android Studio en la versión 3.5.3. Esto se debe a la necesidad de una versión superior a la 3.1 para desarrollar aplicaciones que hagan uso de ARCore. Una vez instalado, crearemos un nuevo proyecto (RABase) con una única actividad, la cual estará formada por un fragmento de Realidad Aumentada y una barra superior cuya funcionalidad será mostrar los botones del menú de la aplicación. Para habilitar el uso de ARCore primero se deben realizar unos cambios en el manifest de la aplicación, añadiendo las entradas AR Required o AR Optional. De esta forma la aplicación necesitará tener instalado obligatoria u opcionalmente los servicios de ARCore antes de poder ejecutarse. Además, se deberá permitir el acceso al hardware de la cámara. Una vez añadidas dichas modificaciones, se agregará la librería de ARCore como dependencia en el build.gradle y se incluirá el repositorio Maven de Google. Por último, la versión mínima del SDK debe ser superior a la 24. Una vez completados todos los cambios, la aplicación ya podrá hacer uso de la librería ARCore.

Para la utilización de Scenform deberemos buscar el plugin en los repositorios de Android Studio e instalarlo. La versión que se utilizará en este proyecto es la 1.15.0. Al igual que en ARCore, se deberá realizar una serie de cambios en el manifest y el build.graddle para el correcto funcionamiento del plugin. Sceneform requiere la librería OpenGL 3.0 y la entrada AR Required en el manifest. Además, hace uso de constructores en Java 8 por lo que si el minSDKVersion es menor a la version 26 se deberá añadir la opción de compilación en el build.gradle. En los códigos 3.1 y 3.2 se pude observar un ejemplo de todo lo explicado anteriormente. La instalación del plugin junto con la versión de Android vienen detallados en el apéndice A.



(a) Informacion de ayuda



(b) Pantalla de ajustes

Figura 3.2: En las siguientes subfiguras se muestran las pantallas tanto de ayuda como de configuración de la aplicacion. La 3.2(a) muestra un mensaje de ayuda al usuario sobre los pasos a seguir para mostrar un objeto 3D . Por último, la 3.2(a) muestra la pantalla de ajustes de la aplicación.

Código 3.1: En esta figura se muestran las líneas de código que se necesitan introducir en el **Manifest** para el correcto funcionamiento de ARCore junto con Sceneform.

```
<uses-permission android:name="android.permission.CAMERA" />

<uses-feature
    android:name="android.hardware.camera.ar"
    android:required="true" />
<uses-feature
    android:glEsVersion="0x00030000"
    android:required="true" />

<application
    ...
    <meta-data
        android:name="com.google.ar.core"
        android:value="required" />
```

Código 3.2: En esta figura se muestran las líneas de código que se necesitan introducir en el **build.graddle** para el correcto funcionamiento de ARCore junto con Sceneform.

```
android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "es.david.redondo.reaumentada"
        minSdkVersion 24
        targetSdkVersion 28
        ...
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    buildTypes {
        ...
    }}

dependencies {
    ...
    // Proporciona la sesión ARCore y recursos relacionados.
    implementation "com.google.ar:core:1.16.0"
    // Proporciona ARFragment
    implementation "com.google.ar.sceneform.ux:sceneform-ux:1.15.0"
    // Uso de ARSeceneView
    implementation "com.google.ar.sceneform:core:1.15.0"
    // Proporciona Animaciones
    implementation "com.google.ar.sceneform:animation:1.15.0"
```

3.3. Creación de Objetos 3D

En esta sección se explicará de la manera más sencilla y simple posible el proceso de modelado que debe seguir un diseñador para la creación de objetos 3D que sean utilizables en la aplicación de forma óptima. En este apartado el programador no tiene por qué ser un experto del diseño 3D ya que en la mayoría de las empresas suele haber gente especializada en este sector. Sin embargo, es recomendable que se posea ciertos conocimientos básicos que pueden ayudar al trabajo en equipo entre diseñadores y programadores. A continuación, se muestran los conceptos básicos antes de comenzar a modelar.

- **Vertices** - Puntos con coordenadas espaciales (x,y,z)
- **Arista** - Linea que une dos vértices
- **Cara o Polígono** - Unión entre 3 o más aristas.
- **Malla** - Conjunto de polígonos para formar un objeto 3D.

3.3.1. Fases de Creación

Se pude diferenciar de forma muy general el proceso de modelado de un objeto 3D en cuatro etapas: Modelado, Texturizado, Animación y Exportación. Para una mejor comprensión del aprendizaje básico del modelado se mostrará la creación de una casa a modo de ejemplo. Para ello se ha utilizado Blender [1] como programa de edición 3D.

Modelado

Para la fase de Modelado se suele hacer uso de programas de diseño en 3D como pueden ser Blender o Cinema 4D [2]. Existen principalmente dos técnicas básicas de diseño dependiendo de si el objeto esta compuesto por una superficie dura o por el contrario, es orgánico. Si nos encontramos en la primera situación la mejor opción es **Box modeling**, ya que se parte de una figura simple (Plano, Cubo, Esfera, etc.) y se van añadiendo polígonos dotando de un mayor detalle al objeto. Se puede observar un ejemplo en la figura 3.3. Otro método también utilizado para este tipo de diseños es el modelado por **NURBS** (non-uniform rational B-spline), en el cual se crea la silueta del objeto que se desea crear y se genera de forma automática la geometría intermedia. El modelado mediante NURBS es mucho menos eficiente ya que no se tiene el control completo de la maya.

En el caso de que se desee crear objetos 3D con cuerpos orgánicos se utiliza la técnica de **Sculpt modeling** la cual simula la escultura de una figura con arcilla. Este método genera objetos de gran detalle con una gran cantidad de polígonos.

[1] https://www.blender.org/ (08/07/2020)
[2] https://www.maxon.net/es/productos/cinema-4d/cinema-4d/ (08/07/2020)

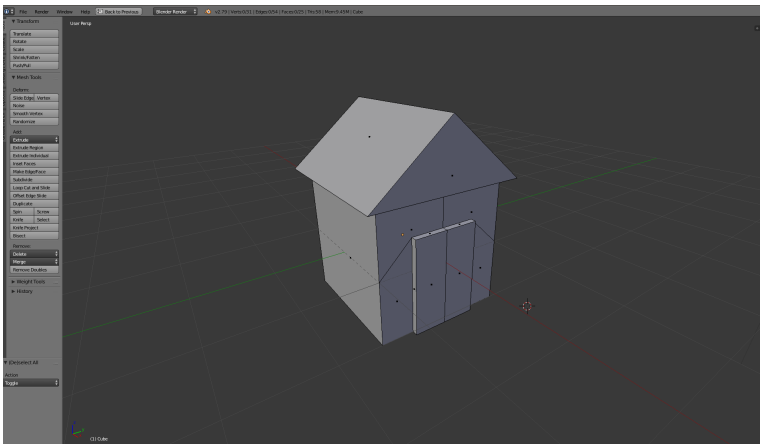


Figura 3.3: Casa modelada mediante Box modeling

Cuando se necesitan objetos de gran detalle con la menor cantidad de polígonos posibles por temas de rendimiento se suele hacer **Retopología**. Esta técnica consiste en generar una malla sobre el objeto en alta densidad con el menor número de polígonos posibles a la cual se le pasan todas las texturas y normales de la malla detallada. Es por ello que la retopología se ha vuelto en una técnica muy importante para sectores como la animación, el desarrollo de videojuegos o la Realidad Aumentada.

Dependiendo de la plataforma en la que se vaya a utilizar el modelo 3D, hay que tener en cuenta el máximo número de polígonos que debe tener la malla para un buen resultado. Para dispositivos móviles los objetos deben contener entre 300 y 1500 polígonos. Para ordenadores entre 1500 y 4000. Los videojuegos de PS3 contienen entre 5000 y 7000 triángulos.

Texturizado

El texturizado es el proceso de dotar a un objeto 3D de materiales y detalles mediante el uso de texturas. Para texturizar un objeto y que se conserve en otros programas de edición 3D se deben crear diferentes mapas.

Mapas UV: Las coordenadas UV permiten colocar texturas 2D en superficies 3D. Es el primer paso en el texturizado ya que permite la conexión entre la malla y la forma en la que se aplicará el mapa de texturas sobre ella.

Mapa de texturas: Es la proyección de una imagen sobre un modelo 3D. Estas imágenes están formadas por píxeles y dependiendo de la resolución que contengan pueden tener mayor o menor detalle. Las texturas con resoluciones más altas requieren de una mayor capacidad de procesamiento de gráficos.

Para que un objeto de bajo poligonaje muestre texturas con relieve y profundidad con gran detalle existen varios tipos de mapeados diferentes que facilitan esta labor.

Mapas de choque: También llamados bumb maps, crean una ilusión de profundidad utilizando un truco de luminosidad. Son imágenes en escala de grises, en los cuales los tonos negros representan las zonas bajas del objeto y las blancas zonas altas.

Mapas de desplazamiento: Similar a los mapas de choque cuya principal diferencia es aportar un relieve real modificando la malla del objeto 3D.

Mapas de normales: Es la evolución de los mapas de choque ya que representan el volumen de una forma más realista. Los mapas de normales son muy coloridos ya que es la combinación de 3 canales. Para el eje X se muestra con el color rojo, para el eje Y con el color verde, y para el eje Z con el color azul.

Mapas de Oclusión Ambiental: Permite dotar al objeto de mayor realismo mediante atenuación de la luz oscureciendo aquellas zonas en las que llegan menos rayos de luminosidad.

Una vez comprendidos los conceptos anteriores se ha procedido a texturizar el objeto 3D que se ha creado anteriormente en la fase de modelado. En la figura 3.4 se puede observar en el lado izquierdo el mapa de texturas y en el derecho el resultado final. Para simplificar todo el proceso se han creado únicamente los mapas de UV y de texturas.

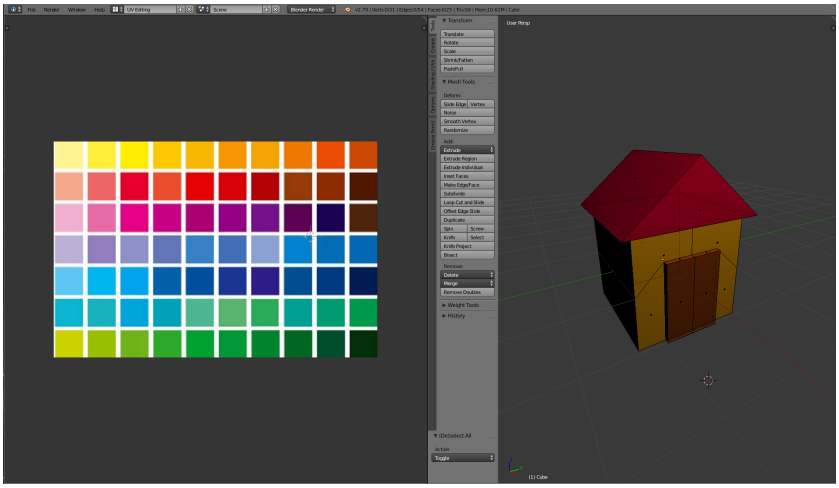


Figura 3.4: Texturizado del modelado de la casa

Animación

Normalmente de esta fase se encargan departamentos especializados en animación ya que supone de una gran complejidad. Cuando se desea añadir animaciones a un objeto 3D, primeramente hay que realizar un proceso de **Rigging**. Esta técnica consiste en ir generando huesos al cuerpo del objeto hasta formar un esqueleto completo el cual se utilizará posteriormente para animar el modelo de la forma más intuitiva y rápida posible.

En este caso para la animación de objetos se ha utilizado la herramienta gratuita de Adobe Mixa-

mo [3], en el cual se permite cargar el objeto deseado y animarlo de forma automática. Esta plataforma permite animar modelos sin tener conocimientos previos con algunas de las animaciones ya existentes en la herramienta. En la aplicación RABase únicamente se ha animado el modelo de la chica (se puede ver reflejado en la figura 3.5) para mostrar como un objeto en formato FBX se puede importar con animaciones en Sceneform.

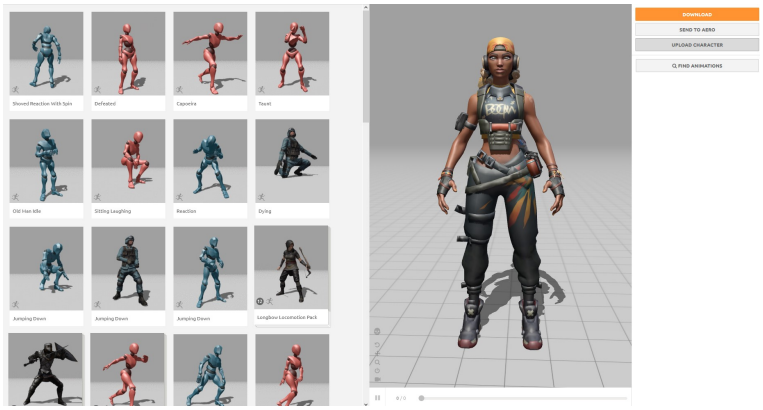


Figura 3.5: Animación del modelo de la chica en Mixamo

Exportación

Una vez se ha finalizado el modelo y se ha texturizado correctamente, se procede a exportar el objeto 3D a un formato deseado. De esta manera se puede transportar objetos 3D entre distintas aplicaciones de edición sin perder información.

3.3.2. Modelos 3D disponibles

En la figura 3.6 se muestran los objetos 3D utilizados en la aplicación de Realidad Aumentada. Todos ellos menos el modelado de la casa han sido descargados (Pelota [4], Trofeo [5], Zanahoria [6], Pato [7], Chica [8]) de páginas gratuitas como Sketchfab ya que el objetivo principal del proyecto no reside en el aprendizaje de modelado 3D sino en el desarrollo de un proyecto con Realidad Aumentada, y como hemos comentado anteriormente hay gente especializada en este sector.

Cada uno de los objetos se encuentran creados en las diferentes extensiones aceptadas por el plugin Sceneform (FBX, OBJ y GLTF) para aprovechar las principales características de cada formato.

[3] https://www.mixamo.com/ (08/07/2020)
[4] https://sketchfab.com/3d-models/soccer-ball-46c91864ef384158b0078e20bdbfe3e9 (08/07/2020)
[5] https://sketchfab.com/3d-models/trophy-2e32184ea51d4b50968beaf48aef0d2a (08/07/2020)
[6] https://sketchfab.com/3d-models/super-carrot-409624f89a75424ea5a2562018d5276f (08/07/2020)
[7] https://sketchfab.com/3d-models/rubber-duck-a84cecb600c04eeba60d02f99b8b154b (08/07/2020)
[8] https://sketchfab.com/3d-models/raze-valorant-6eddd402eb2b4535ab4ec9cb5d1983e5 (08/07/2020)

Figura 3.6: Modelos disponibles en la aplicación RABase

3.4. Implementación

Una vez el proyecto esté listo para utilizar ARCore junto con Sceneform y se tengan los objetos 3D deseados, se procederá a desarrollar la aplicación. Para ello hay que importar los modelos dentro de Android Studio haciendo uso del plugin. Sceneform trabaja con sus propios formatos de objeto por lo que en el proceso de importación se generan modelos 3D en SFA y SFB para poder usarlos durante la ejecución. SFA se trata de un archivo precompilado utilizado para realizar cambios en el modelo 3D y SFB es el archivo binario usado en la renderización del objeto. Este proceso de importación viene explicado en el apéndice C. Los objetos en formato GLTF y GLTB se pueden utilizar directamente.

En la figura 3.7 se muestra un esquema básico de la estructura global de la aplicación. Se trata de una arquitectura de aplicación clásica compuesta de actividades y fragmentos. Esto permite introducir la Realidad Aumentada en otras aplicaciones añadiendo el fragmento creado con sus respectivas dependencias. A continuación, se explicará la implementación de la aplicación RABase haciendo uso de Java como lenguaje de programación.

3.4.1. MainActivity

Es la actividad principal de la aplicación. En ella se cargará el fragmento CustomArFragment de Realidad Aumentada junto con los controladores de los botones del menú superior. Como se puede comprobar en el código 3.3, cuando se ejecuta la función **onResume()** se selecciona el tipo de reconocimiento que se va a utilizar según las preferencias guardadas en el sistema.

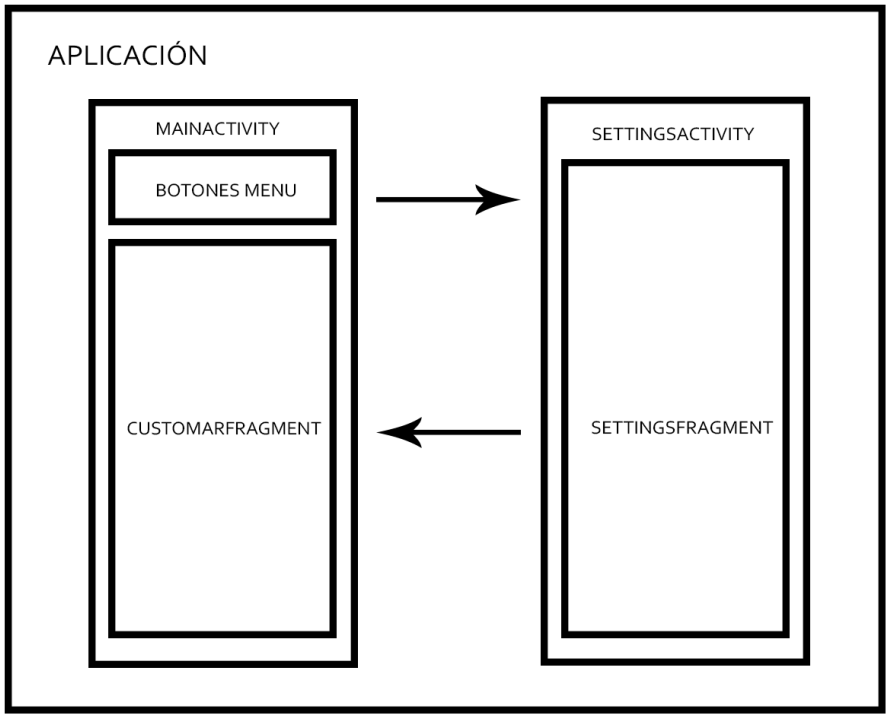


Figura 3.7: Arquitectura global del sistema

Código 3.3: En esta figura se muestran las líneas de código en las que se selecciona el tipo de reconocimiento que se va a utilizar. Si se cumple la condición se ejecutará el codigo relacionado con el restreo de imagenes. En caso contrario se ejecutará el codigo relacionado con el rastreo de planos.

```
1   @Override
2   protected void onResume() {
3       super.onResume();
4
5       if(prefs.getBoolean("tipo_reconocimiento",false)==false){
6           Log.i("DEBUG","RESUME-AugmentedImage");
7           arFragment.getArSceneView().getScene().addOnUpdateListener(this::onUpdateFrame);
8
9       }else{
10          Log.i("DEBUG","RESUME-Actividad_Plano");
11          arFragment.setOnTapArPlaneListener((hitResult, plane, motionEvent) -> {
12              createModel(hitResult.createAnchor(),arFragment);
13          });
14      }
```

Rastreo de Planos

Cuando un usuario pulse en cualquier punto del plano detectado se ejecutará la función **createModel()** la cual viene mostrada en el código 3.4. Lo primero de todo es obtener de las preferencias, el objeto que se desea crear. Una vez cargada la selección, se procede a construir el modelo renderizable creando como punto de anclaje (AnchorNode) la coordenada en el plano en la que el usuario ha pulsado la pantalla. A continuación, se crea un Nodo esqueleto (SkeletonNode), el cual es necesario para la reproducción de animaciones. Una vez creado el punto de anclaje y el nodo esqueleto se procede a dotar al modelo de posición, rotación, sombras, etc. Finalmente se añade la dependencia del punto de anclaje con la escena mostrando el avatar en la pantalla.

Código 3.4: En esta figura se muestran las líneas de código correspondientes con la creación de un objeto 3D cuando se pulsa en un punto determinado del plano.

```
1  private void createModel(Anchor anchor, ArFragment arFragment){
2
3      String modelo = prefs.getString("listado_modelos","");
4
5      ModelRenderable
6              .builder()
7              .setSource(this, Uri.parse(modelo))
8              .build()
9              .thenAccept(modelRenderable -> {
10
11                 //SOMBRAS DEL OBJETO
12                 modelRenderable.setShadowCaster(prefs.getBoolean("sombras_objeto",false));
13                 modelRenderable.setShadowReceiver(prefs.getBoolean("sombras_objeto",false));
14
15                 AnchorNode anchorNode = new AnchorNode(anchor);
16                 SkeletonNode skeletonNode = new SkeletonNode();
17                 skeletonNode.setParent(anchorNode);
18                 Pose pose = Pose.makeTranslation(0.0f,0.0f,0.0f);
19                 skeletonNode.setLocalPosition(new Vector3(pose.tx(),pose.ty(),pose.tz()));
20                 skeletonNode.setLocalRotation(new
                       Quaternion(pose.qx(),pose.qy()+180,pose.qz(),pose.qw()));
21                 skeletonNode.setRenderable(modelRenderable);
22
23                 arFragment.getArSceneView().getScene().addChild(anchorNode);
24
25                 Button button = findViewById(R.id.play_button);
26                 button.setOnClickListener(view -> animateModel(modelRenderable));
27
28             });
29 }
```

Rastreo de Imagenes

El usuario deberá buscar la imagen establecida para la creación del objeto. El sistema irá analizando cada frame en busca de dicha imagen hasta que sea capaz de rastrearla haciendo uso de la función **onUpdateFrame()** mostrada en el código 3.5.

Cada frame se puede encontrar en cualquiera de los siguientes 3 estados: **PAUSED**, **TRACKING** y **STOPPED**.

PAUSED: Se produce cuando ARCore ha detenido el seguimiento de la instancia, aunque puede reanudarlo en el futuro. Cuando se encuentra en este estado, las propiedades de la instancia pueden ser muy imprecisas por lo que generalmente no deben utilizarse para generar los avatares.

TRACKING: Se produce cuando la imagen es rastreable y su posición es actual. En este estado será donde se cree el objeto 3D que se desea renderizar. Para ello se creará un Nodo personalizado el cual se explicará más adelante (3.4.2 MyARNode). A continuación, se introducirá la imagen rastreada en el nodo creado y se asociará con el avatar renderizado mediante un HashMap. De esta manera podremos tener varios objetos asociados a diferentes imágenes de forma simultánea.

STOPPED: Se produce cuando ARCore deja de rastrear la imagen y no volverá a hacerlo. Cuando la aplicación alcanza este estado se elimina la imagen del HashMap creado anteriormente.

Para optimizar el rastreo de imágenes existe la herramienta **arcoreimg** la cual analiza una imagen y le otorga una puntuación entre 0 y 100 para su utilización en ARCore. Soporta tanto los formatos JPG como PNG y la resolución mínima debe ser de 300 x 300 pixeles. Algunos de los consejos para un mejor rendimiento consisten en evitar imágenes repetitivas y con características escasas.

Reproducción de animaciones

Para la reproducción de animaciones se hace uso de la funcionalidad mostrada en el código 3.6. Lo primero que se realiza es una comprobación de si existe un modelAnimator y si este se encuentra en reproducción. En el caso de que se cumpla la condición se finalizará la reproducción de la animación actual y se procederá a reproducir la siguiente de la lista.

Código 3.5: En esta figura se muestran las líneas de código correspondientes a la creación de un objeto 3D mediante el rastreo de imagenes

```
1  public void onUpdateFrame(FrameTime frameTime) {
2      Frame frame = arFragment.getArSceneView().getArFrame();
3
4      if(frame==null){
5          return;
6      }
7
8      Collection<AugmentedImage> updateAugmentedImg =
           frame.getUpdatedTrackables(AugmentedImage.class);
9
10     for(AugmentedImage image:updateAugmentedImg){
11         switch (image.getTrackingState()){
12             case PAUSED:
13                 Log.i("DEBUG","Imagen_detectada");
14                 break;
15             case TRACKING:
16                 if (!augmentedImageMap.containsKey(image)) {
17                     Log.i("DEBUG","NUEVO_NODO_AUGMENTD_IMAGES");
18                     node = new MyARNode(this);
19                     node.setImage(image);
20                     augmentedImageMap.put(image, node);
21                     arFragment.getArSceneView().getScene().addChild(node);
22                 }
23                 break;
24             case STOPPED:
25                 augmentedImageMap.remove(image);
26                 break;
27         }
28     }
29 }
```

Código 3.6: En esta figura se muestran las líneas de código correspondientes a la reproducción de las animaciones que contenga un objeto.

```
1   private void animateModel(ModelRenderable modelRenderable){
2       if (modelAnimator != null && modelAnimator.isRunning())
3           modelAnimator.end();
4
5       int animationCount = modelRenderable.getAnimationDataCount();
6
7       if(i==animationCount){
8           i=0;
9           return;
10      }
11
12      AnimationData animationData =modelRenderable.getAnimationData(i);
13
14      modelAnimator = new ModelAnimator(animationData,modelRenderable);
15      modelAnimator.start();
16
17      i++;
18  }
```

3.4.2. MyARNode

Esta clase es una modificación de un Nodo de Anclaje ya que hereda de la clase AnchorNode. Cuando se crea una instancia de esta clase se creará un modelo renderizable. Sin embargo, el objeto no se mostrará en la pantalla hasta que se le introduzca la imagen rastreada. Si nos fijamos en el código 3.8, al igual que en el rastreo de planos, se dota al objeto de posición, rotación, etc. Además, se observa la ausencia de sentencias de creación de nodos esqueleto. Esto se debe a la decisión de generar objetos sin cargar las animaciones a modo de muestra para aquellos objetos que no posean. Cuando se produce un error durante el procedimiento de asociación de una imagen con el objeto 3D creado anteriormente se lanzará una excepción junto con un mensaje de texto.

Código 3.7: En esta figura se muestran las líneas de código correspondientes al constructor MyARNode

```
1   public MyARNode(Context context){
2       SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(context);
3       String modelo = prefs.getString("listado_modelos","");
4
5       if(modelRenderable ==null){
6           modelRenderable = ModelRenderable.builder()
7                   .setSource(context, Uri.parse(modelo))
8                   .build();
9       }
10  }
```

Código 3.8: En esta figura se muestran las líneas de código correspondientes a la introducción de una imagen rastreada y el posterior renderizado del modelo 3D.

```
1   public void setImage(AugmentedImage imagen){
2       this.imagen = imagen;
3       if(!modelRenderable.isDone()){
4           CompletableFuture.allOf(modelRenderable)
5                   .thenAccept((Void aVoid)->{
6                       setImage(imagen);
7                   }).exceptionally(throwable -> {
8               Log.e(TAG, "Error_cargando", throwable);
9                       return null;
10          });
11      }
12
13
14      setAnchor(imagen.createAnchor(imagen.getCenterPose()));
15      Node node = new Node();
16      Pose pose = Pose.makeTranslation(0.0f,0.0f,0.0f);
17      node.setParent(this);
18      node.setLocalPosition(new Vector3(pose.tx(),pose.ty(),pose.tz()));
19      node.setLocalRotation(new Quaternion(pose.qx(),pose.qy(),pose.qz(),pose.qw()));
20      node.setRenderable(modelRenderable.getNow(null));
21
22  }
```

3.4.3. CustomArFragment

Esta clase es la encargada de crear un fragmento de Realidad Aumentada de forma personalizada puesto que hereda de ArFragment. En ella se comprobará tanto la versión Android del dispositivo como de la librería OpenGL. En el caso de que no se cumpla alguno de los requisitos se mostrará un mensaje de error.

En la función **getSessionConfiguration()** mostrada en el código 3.9 se llevan a cabo todos los ajustes de la sesión de Realidad Aumentada disponibles en la aplicación.

Modo de Autoenfoque

Existen dos posibles enfoques de cámara: **AUTO** Y **FIXED**. Esta configuración se muestra en el código 3.10.

AUTO: El modo auto dota a la aplicación de mayor precisión y claridad en tareas de rastreo y reconocimiento de cuerpos perdiendo estabilidad en el sistema a cambio de una experiencia más realista.

FIXED: El modo fixed fija el enfoque de la cámara ganando rendimiento a cambio de un peor detalle.

Código 3.9: En esta figura se muestran las líneas de código correspondientes a la comprobación de versiones de Android y la librería gráfica OpenGl

```
1   @Override
2   public void onAttach(Context context) {
3       super.onAttach(context);
4
5       prefs = PreferenceManager.getDefaultSharedPreferences(getContext());
6
7       if (Build.VERSION.SDK_INT < Build.VERSION_CODES.N) {
8           Log.e(TAG, "Sceneform requiere Android N o superior");
9           SnackbarHelper.getInstance()
10                  .showError(getActivity(), "Sceneform requiere Android N o superior");
11      }
12
13      String openGlVersionString =
14              ((ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE))
15                      .getDeviceConfigurationInfo()
16                      .getGlEsVersion();
17
18      if (Double.parseDouble(openGlVersionString) < MIN_OPENGL_VERSION) {
19          Log.e(TAG, "Sceneform requiere OpenGL ES 3.0 o superior");
20          SnackbarHelper.getInstance()
21                  .showError(getActivity(), "Sceneform requiere OpenGL ES 3.0 o superior");
22      }
23  }
```

Código 3.10: En esta figura se muestran las líneas de código correspondientes a la configuración del enfoque de la cámara.

```
1       //AUTOENFOQUE DE LA CAMARA
2       if(prefs.getBoolean("autoenfoque",false)){
3           config.setFocusMode(Config.FocusMode.AUTO);
4       }else{
5           config.setFocusMode(Config.FocusMode.FIXED);
6       }
```

Modo de Estimación de Luz

La estimación de luz permite aplicar la iluminación del entorno real a los modelos 3D renderizados. Gracias a esta característica se alcanza un mayor realismo. Existen 3 configuraciones: **DISABLED**, **AMBIENT INTENSITY** Y **ENVIRONMENTAL HDR**. Esta configuración se muestra en el código 3.11.

DISABLED: La estimación de la iluminación está desactivada.

AMBIENT INTENSITY: Se habilita la estimación de la iluminación, generando una estimación de intensidad de un solo valor y tres valores de corrección de color (R, G, B)

ENVIRONMENTAL HDR: Se habilita esta opción, generando una estimación de la iluminación HDR ambiental inferida en el espacio de color lineal.

Este modo es incompatible con la cámara frontal (selfie). Si se establece en una sesión creada para la cámara frontal, la llamada a configurar fallará.

Código 3.11: En esta figura se muestran las líneas de código correspondientes a la configuración de la estimación de luz.

```
1        //ILUMINACION
2        switch (prefs.getString("listado_iluminacion","1")){
3            case "1":
4                config.setLightEstimationMode(Config.LightEstimationMode.DISABLED);
5            case "2":
6                config.setLightEstimationMode(Config.LightEstimationMode.AMBIENT_INTENSITY);
7            case "3":
8                config.setLightEstimationMode(Config.LightEstimationMode.ENVIRONMENTAL_HDR);
9        }
```

Database Augmented Image

En la función **setupAugmentedImageDatabase()** mostrada en el código 3.12, se crea la base de datos en la cual se introducirán las imágenes que se van a utilizar posteriormente en el rastreo de imágenes. Si se cumple la condición de utilizar una imagen en formato png o jpg (USE_SINGLE_IMAGE) se transformará el archivo en un mapa de bits para su posterior introducción en la base de datos. En caso contrario se generará la base de datos con la imagen ya precargada en formato imdb.

En la función **loadAugmentedImageBitmap()** mostrada en el código 3.12 se transforma la imagen deseada en un mapa de bits. Esto es necesario para poder añadir la imagen en la base de datos.

3.4.4. SettingsActivity

Actividad encargada de la gestión de los ajustes de la aplicación. Para su generación se utiliza un fragmento modificado, el cual se cargan las preferencias desde el recurso R.xml.preferences. Desde

Código 3.12: En esta figura se muestran las líneas de código correspondientes a la creación de la base de datos para imagenes

```
1  private boolean setupAugmentedImageDatabase(Config config, Session session) {
2      AugmentedImageDatabase augmentedImageDatabase;
3      AssetManager assetManager = getContext() != null ? getContext().getAssets() : null;
4      if (assetManager == null) {
5          Log.e(TAG, "Context␣null,␣no␣se␣puede␣inicializar␣la␣base␣de␣datos.");
6          return false;
7      }
8
9      if (USE_SINGLE_IMAGE) {
10         Bitmap augmentedImageBitmap = loadAugmentedImageBitmap(assetManager);
11         if (augmentedImageBitmap == null) {
12             return false;
13         }
14
15         augmentedImageDatabase = new AugmentedImageDatabase(session);
16         augmentedImageDatabase.addImage(DEFAULT_IMAGE_NAME, augmentedImageBitmap);
17
18     } else {
19         try (InputStream is = getContext().getAssets().open(SAMPLE_IMAGE_DATABASE)) {
20             augmentedImageDatabase = AugmentedImageDatabase.deserialize(session, is);
21         } catch (IOException e) {
22             Log.e(TAG, "IO␣excepcion␣cargando␣augmented␣image␣bitmap.", e);
23             return false;
24         }
25     }
26     config.setAugmentedImageDatabase(augmentedImageDatabase);
27     return true;
28 }
29
30 private Bitmap loadAugmentedImageBitmap(AssetManager assetManager) {
31     try (InputStream is = assetManager.open(DEFAULT_IMAGE_NAME)) {
32         return BitmapFactory.decodeStream(is);
33     } catch (IOException e) {
34         Log.e(TAG, "IO␣excepcion␣cargando␣augmented␣image␣bitmap.", e);
35     }
36     return null;
37 }
```

aquí se puede configurar el método de rastreo (plano o imagen), el diseño que se desea generar, autoenfoque, sombras, visualización del plano detectado, etc.

La creación de esta actividad resulta muy sencilla puesto que el propio Android Studio ofrece una plantilla que es personalizable. La creación de dicha actividad viene reflejada en el código 3.13 y 3.14.

Código 3.13: En esta figura se muestran las líneas de código correspondientes a la creación de la actividad SettingsActivity

```
1   @Override
2   protected void onCreate(Bundle savedInstanceState) {
3       super.onCreate(savedInstanceState);
4
5       setContentView(R.layout.settings_activity);
6
7       if(findViewById(R.id.fragment_container)!=null){
8           if(savedInstanceState!=null){
9               return;
10          }
11
12          getFragmentManager().beginTransaction().add(R.id.fragment_container,new
                SettingsFragment()).commit();
13      }
```

Código 3.14: En esta figura se muestran las líneas de código correspondientes a la creación del fragmento SettingsFragment

```
1   public class SettingsFragment extends PreferenceFragment {
2       @Override
3       public void onCreate(@Nullable Bundle savedInstance){
4           super.onCreate(savedInstance);
5           addPreferencesFromResource(R.xml.preferences);
6
7       }
```

4

Pruebas

En este capítulo se explicarán todas las pruebas realizadas durante el desarrollo del proyecto desde su fase de creación hasta las pruebas finales en el prototipo.

La aplicación se decidió en un primer momento desarrollarse en Android Studio 3.6 junto con la versión 1.15.0 de Sceneform. Sin embargo, se ha tenido cambiar la version de Android Studio a la 3.5.3 al no funcionar correctamente la importación de objetos.

Una vez conseguido que funcionen correctamente las librerías, se ha procedido a importar diversos objetos con los múltiples formatos y así poder analizar las ventajas de cada uno de ellos. Se ha podido comprobar que el plugin funciona correctamente. Sin embargo, la previsualización de algunos modelos 3D en formato FBX muestran errores de representación en sus texturas. Esto no ha supuesto un impedimento en el desarrollo puesto que en la aplicación todos los objetos se renderizan correctamente.

Se ha probado a generar objetos 3D utilizando los dos métodos implementados, tanto el rastreo de planos como el rastreo de imágenes y ambas funcionan correctamente. Sin embargo, ambas han mostrado dificultades en algunas ocasiones para detectar la superficie sobre la que se puede generar el avatar ya sea el plano o la imagen rastreable.

Se ha hecho uso de la herramienta arcorimg para analizar la optimización de la fotografía utilizada en el rastreo de imágenes. Se ha seleccionado una imagen de la aplicación de matemáticas para niños con TEA [1] ya que el principal objetivo del proyecto es poder integrarlo en un futuro como un módulo auxiliar. La puntuación obtenida ha sido de 100, siendo el máximo posible. Por lo tanto, se ha decidido usarla en la aplicación como ejemplo de rastreo debido a su alta facilidad de detección para la cámara del dispositivo.

Se ha generado un mismo objeto 3D con las diferentes estimaciones de luz posibles sin llegar a detectar ninguna diferencia significativa entre ellas. La estimación de luz se puede ver afectada por la luminosidad del entorno real, propiedades de las texturas del avatar, configuración de enfoque, etc. Por dichos motivos, no se ha podido explorar a fondo esta característica.

Se han realizado pruebas con variaciones en los ajustes de enfoque para comprobar el correcto funcionamiento de la aplicación y analizar si presentan diferencias para la experiencia del usuario. Si la

aplicación presenta dificultades para detectar un plano o una imagen es recomendable utilizar el modo de enfoque AUTO. Este ajuste supone un incremento en los requisitos hardware del dispositivo por lo que se recomienda dejar en FIXED si se presentan problemas de rendimiento. Se pueden observar las diferencias entre estas dos configuraciones en la figura 4.1.

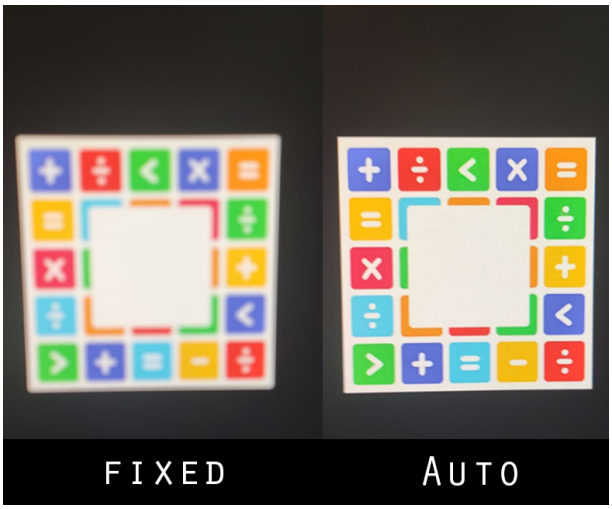
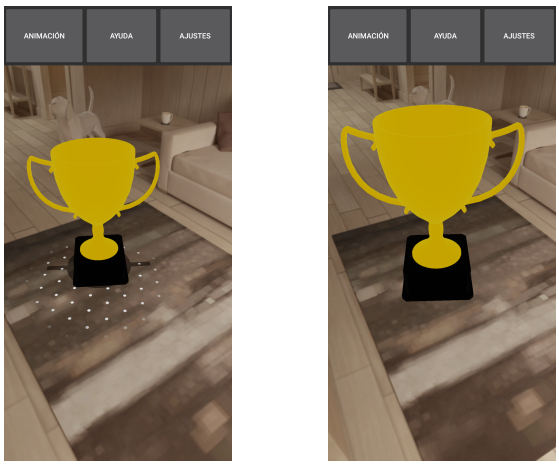


Figura 4.1: Diferencia de enfoque de cámara entre los modos AUTO y FIXED

Para la generación de sombras y visualización del plano se ha procedido a generar un mismo objeto 3D tanto con las opciones activadas como desactivadas. Se puede observar en la figura 4.2 la diferencia entre ambas configuraciones. Se recomienda dejar activo la opción de mostrar plano para detectar más fácilmente las superficies válidas de renderizado.



(a) Sombras y plano activados

(b) Sombras y plano desactivados

Figura 4.2: En las siguientes subfiguras se muestran los diferentes métodos de representación de un avatar. La 4.2(a) es un ejemplo de renderizado de un objeto con sombras en un plano renderizable visible . La 4.2(b) es una muestra de renderizado de un objeto sin sombras en un plano renderizable oculto.

Se han realizado pruebas tanto en dispositivos físicos como en el emulador de Android Studio. La instalación del emulador ha sido detallada en el apéndice B.

Dispositivos Físicos

En el tabla 4.1 se muestran los dispositvos móviles utilizados en la realización de las pruebas.

	Version Android	**Memoria RAM**	**Procesador**
OnePlus5	9	6GB	Qualcomm® Snapdragon™ 835
Xiaomi Mi 9T	10	6GB	Qualcomm® Kryo™ 470
Nexus 5X	8.1	2GB	Qualcomm® Snapdragon 808 MSM8992

Tabla 4.1: Tabla comparativa de los diferentes dispositivos físicos utilizados

Se ha comprobado que la aplicación funciona correctamente en todos los dispositivos que han sido utilizados. Sin embargo, el Nexus 5X ha mostrado problemas de rendimiento al tratarse de un dispositivo móvil más desfasado y con peores prestaciones hardware.

Dispositivos Emulados

En el tabla 4.2 se muestran los dispositivos emulados para la realización de las pruebas.

	Version Android
Pixel 3A XL	10
Pixel 2	10

Tabla 4.2: Tabla comparativa de los diferentes dispositivos emulados utilizados

La aplicación funciona correctamente en todos los dispositivos que han sido emulados. Sin embargo, estas pruebas pueden sufrir variaciones en la realidad ya que se trata de un entorno virtualizado, el cual depende de la potencia suministrada por el ordenador que lo emula.

5

Conclusiones

5.1. Conclusiones

Para este trabajo de fin de grado se propuso realizar un estudio de la Realidad Aumentada. Para ello en el capítulo 3 se han analizado las plataformas existentes en el mercado explicando las características principales de cada una ellas. Una vez seleccionada la herramienta deseada y que mejor se ajusta al desarrollo del proyecto, en el capítulo 4 se ha procedido a implementar una aplicación de prueba (RABase) la cual se encuentra desarrollada con una arquitectura clásica compuesta de actividades y fragmentos lo que le permite ser reutilizable en otros proyectos. Por último, en el apartado 5 han mostrado todas las pruebas realizadas durante el desarrollo del proyecto desde sus fases iniciales hasta su prototipo final.

Aunque la aplicación es reutilizable, el prototipo está planteado para poderse integrar en un futuro en la aplicación de matemáticas para niños con TEA [1] ya existente. Este módulo servirá para recompensar a los alumnos a modo de motivación cuando resuelvan los ejercicios correctamente. Es por ello por lo que se han utilizado objetos simples e imágenes ya existentes.

El uso de la librería ARCore junto con el plugin Sceneform ha resultado ser muy satisfactoria ya que no es necesario tener amplios conocimientos de la librería OpenGL ni de modelado 3D. Sin embargo, esta simplicidad en la implementación se ha visto afectada en el rendimiento ya que en determinadas ocasiones el sistema de rastreo no ha funcionado de la manera esperada. A pesar de ello, la herramienta funciona correctamente en la mayoría de las situaciones. Otro aspecto positivo de las librerías es su constante estado de actualización y mejora que permite solventar los errores producidos en versiones anteriores.

Como conclusión se puede asegurar que la Realidad Aumentada tiene un gran potencial para el futuro. Sin embargo, para el caso de Android, se han observado limitaciones en las herramientas de desarrollo a comparación de otras plataformas más especializadas en tratamiento de objetos 3D como pueden ser Unity o Unreal Engine. Esto se debe al tratarse de herramientas muy novedosas que se encuentran en fases muy tempranas de desarrollo. A pesar de ello, los resultados obtenidos han sido satisfactorios cumpliendo sobradamente las necesidades del proyecto permitiendo observar

los posibles beneficios que puede alcanzar esta tecnología en niños con TEA permitiéndoles explorar nuevas técnicas de aprendizaje.

5.2. Trabajo Futuro

En base al trabajo realizado en este proyecto, aparecen nuevas rutas de investigación que permiten explorar los posibles beneficios de la Realidad Aumentada en el sector educativo, concretamente en niños con TEA. Los trabajos futuros considerados son los siguientes:

- Integrar la funcionalidad desarrollada en este proyecto en la aplicación de matemáticas para niños con TEA [1]. Dicho módulo permitirá premiar a los alumnos con objetos visuales e interactivos al realizar las correcciones de los ejercicios satisfactoriamente.
- Evaluar la aplicación final con alumnos con TEA y realizar un estudio sobre el grado de aceptación por parte de los usuarios a esta herramienta.

Bibliografía

[1] J. C. Torrado, J. Gomez, and L. Jaccheri, "Supporting self-evaluation for children with mental disabilities through augmented reality," in *Proceedings of the 18th ACM International Conference on Interaction Design and Children*, IDC '19, (New York, NY, USA), p. 635–641, Association for Computing Machinery, 2019.

[2] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Computer Graphics and Applications*, vol. 21, no. 6, pp. 34–47, 2001.

[3] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE TRANSACTIONS on Information and Systems*, vol. 77, no. 12, pp. 1321–1329, 1994.

[4] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.

[5] "Easyar-best engine for developing augmented reality." `https://www.easyar.com/`. (Último acceso 08/07/2020).

[6] "Opengl - the industry standard for high performance graphics." `https://www.opengl.org/`. (Último acceso 08/07/2020).

[7] "Spark ar studio: crea experiencias de realidad aumentada | spark ar studio." `https://sparkar.facebook.com/ar-studio/`. (Último acceso 08/07/2020).

[8] "Vuforia developer portal |." `https://developer.vuforia.com/`. (Último acceso 08/07/2020).

[9] "Build new augmented reality experiences that seamlessly blend the digital and physical worlds." `https://developers.google.com/ar`. (Último acceso 08/07/2020).

[10] "Sceneform overview | sceneform (1.15.0) | google developers." `https://developers.google.com/sceneform/develop`. (Último acceso 08/07/2020).

Apéndices

A

Creación del Proyecto

A.1. Versión de Android Studio

Figura A.1: Versión de Android Studio utilizada en el desarrollo del proyecto

A.2. Instalación de Sceneform

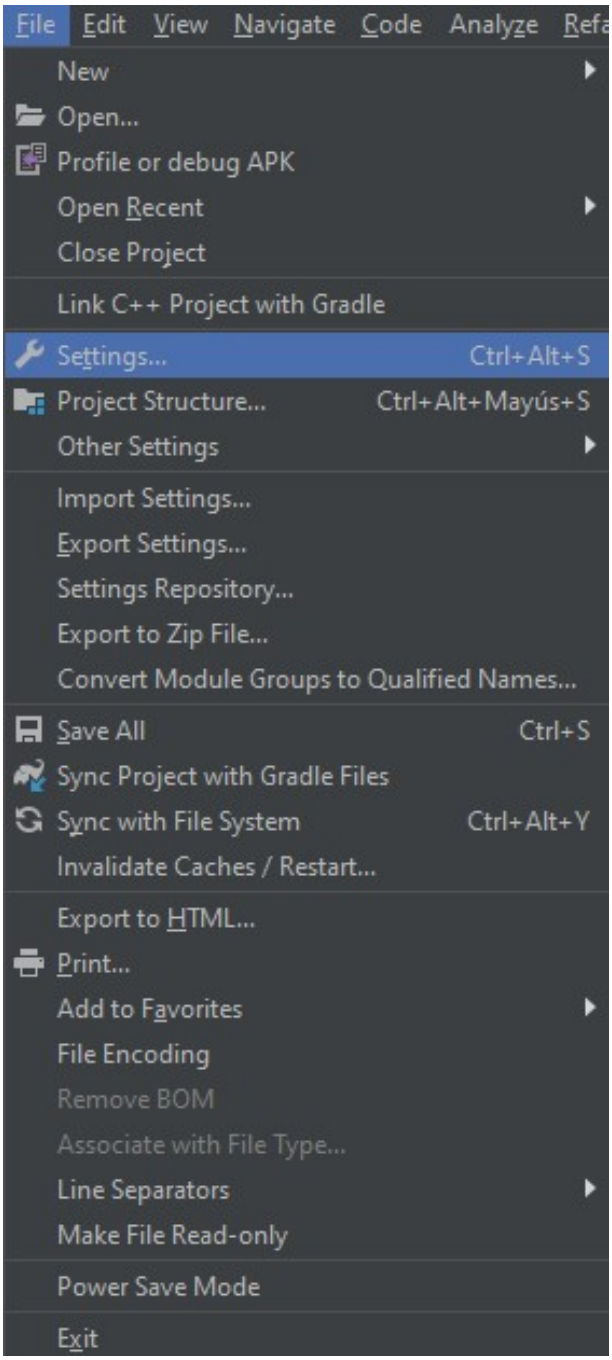


Figura A.2: Buscamos en la pestaña File la opción Settings

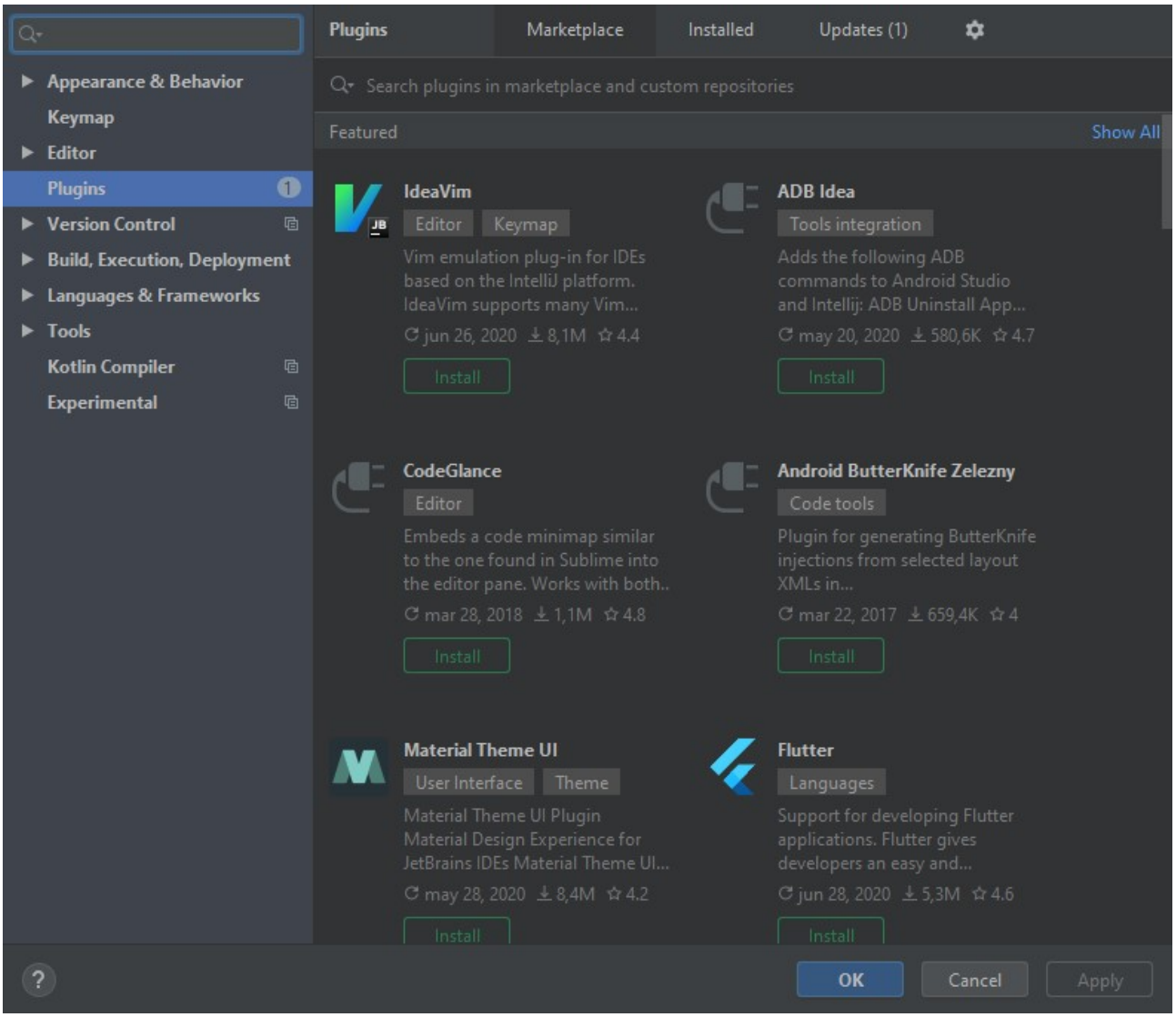


Figura A.3: Buscamos en la pestaña File la opción Settings

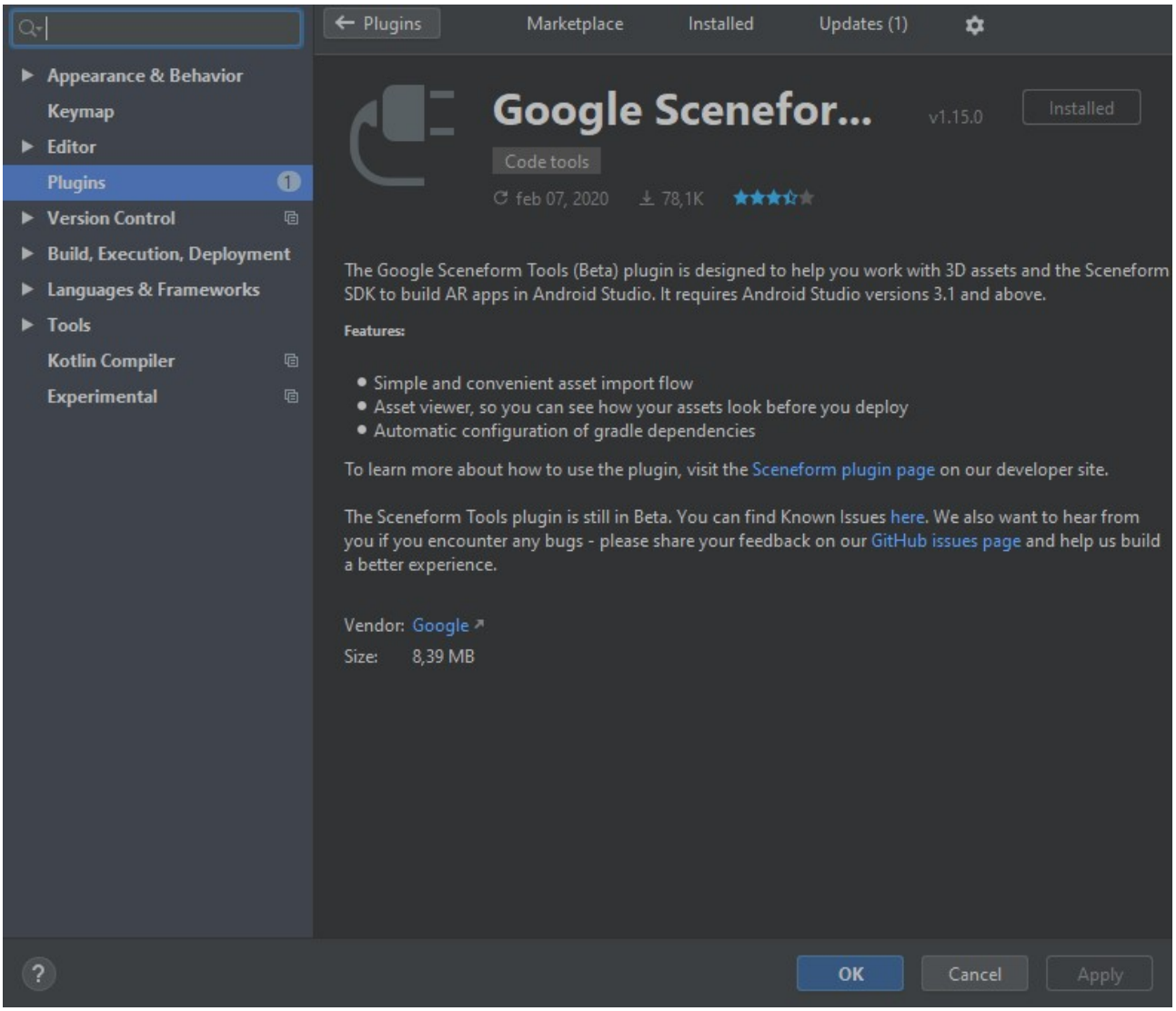


Figura A.4: Pulsamos sobre la opción Install y ya se dispondrá de la librería para su utilización.

B

Instalación del emulador de Android Studio

B.1. Creación del emulador

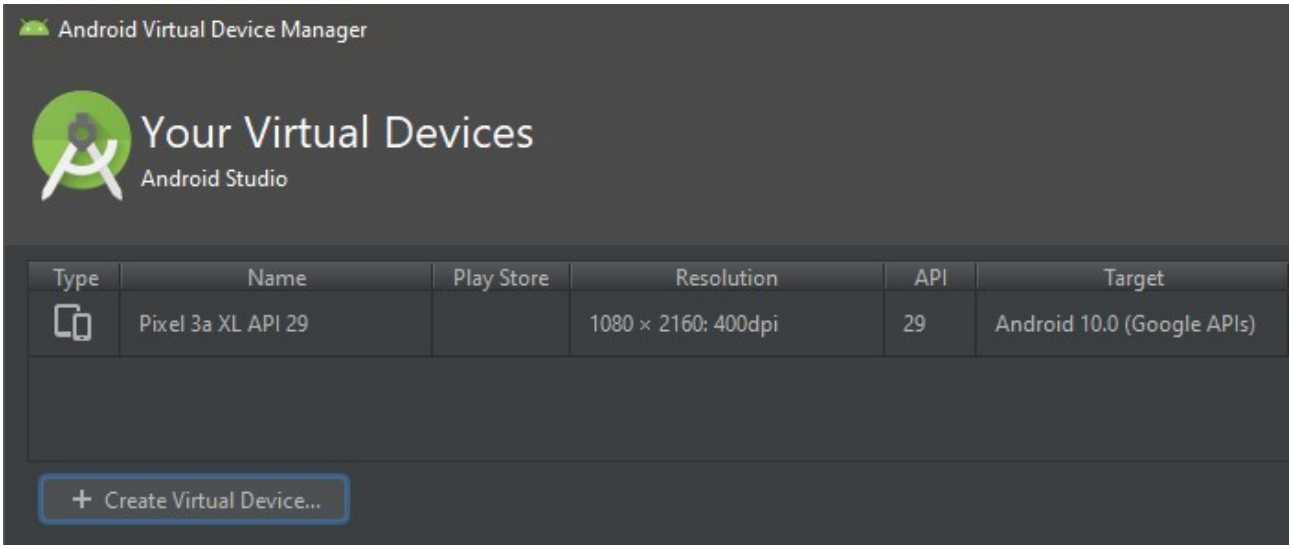


Figura B.1: Pulsamos sobre la opcion Create Virtual Device

B.2. Selección del hardware

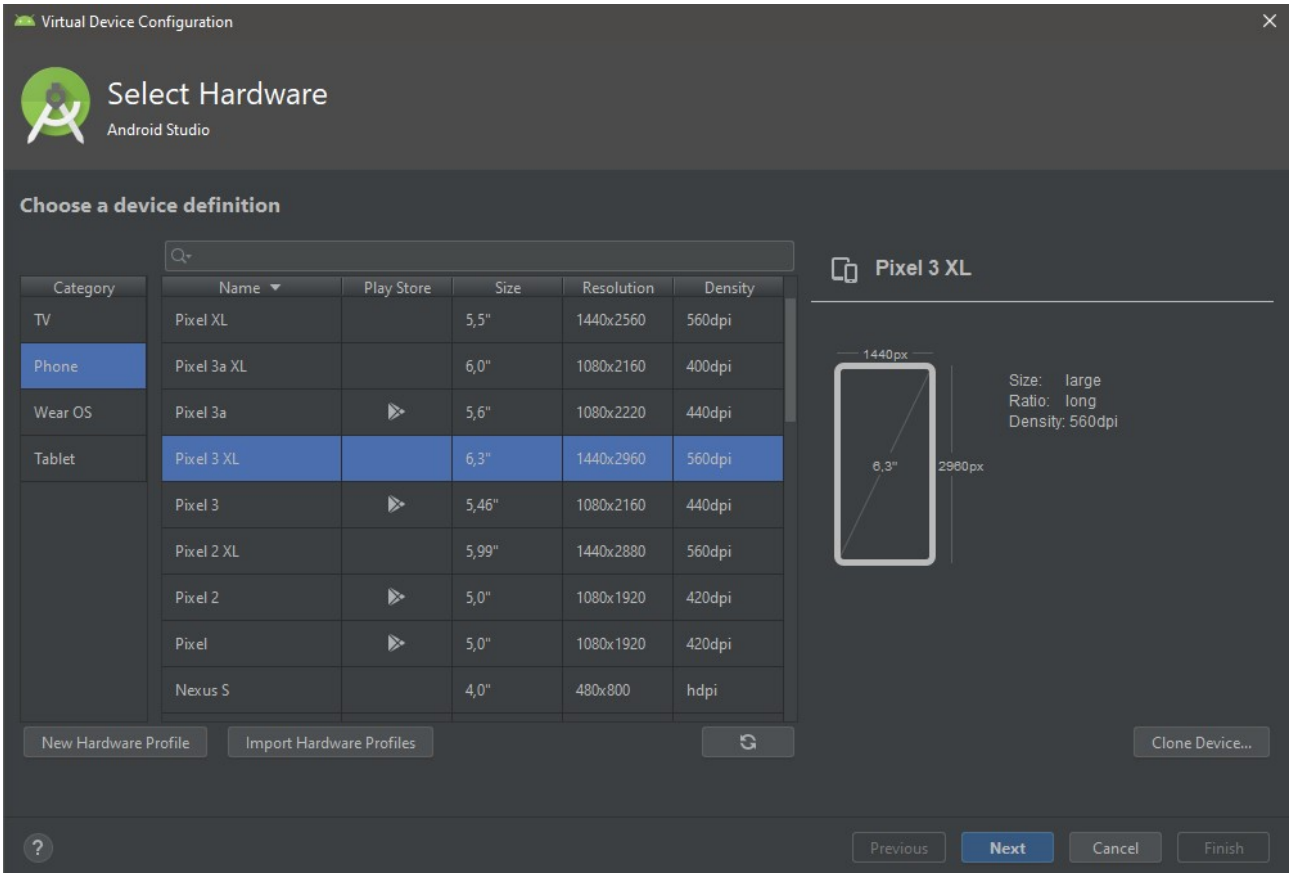


Figura B.2: Seleccionamos el dispositivo que deseamos emular

B.3. Selección de la versión Android

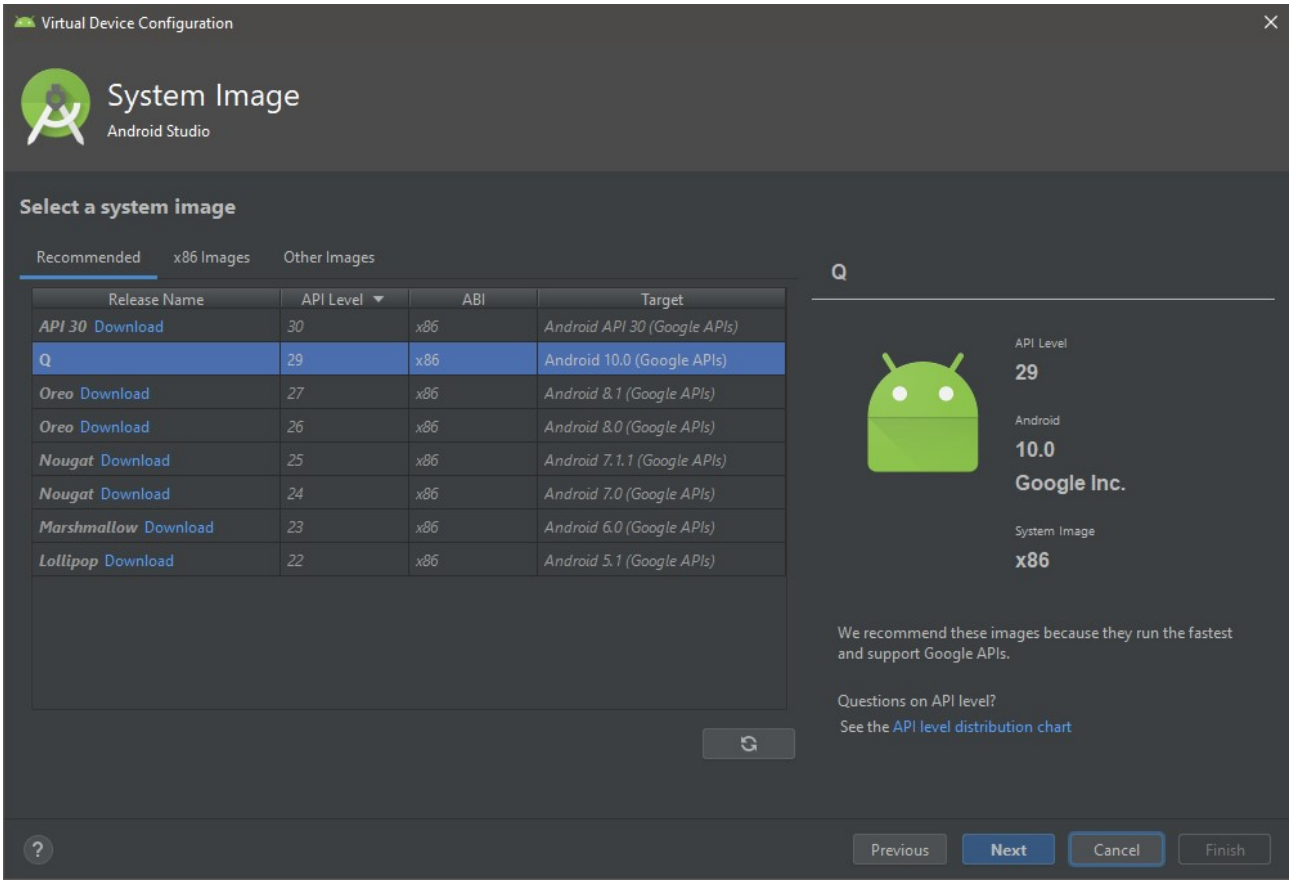


Figura B.3: Seleccionamos la versión de Android que vamos a utilizar

B.4. Verificación de Instalación

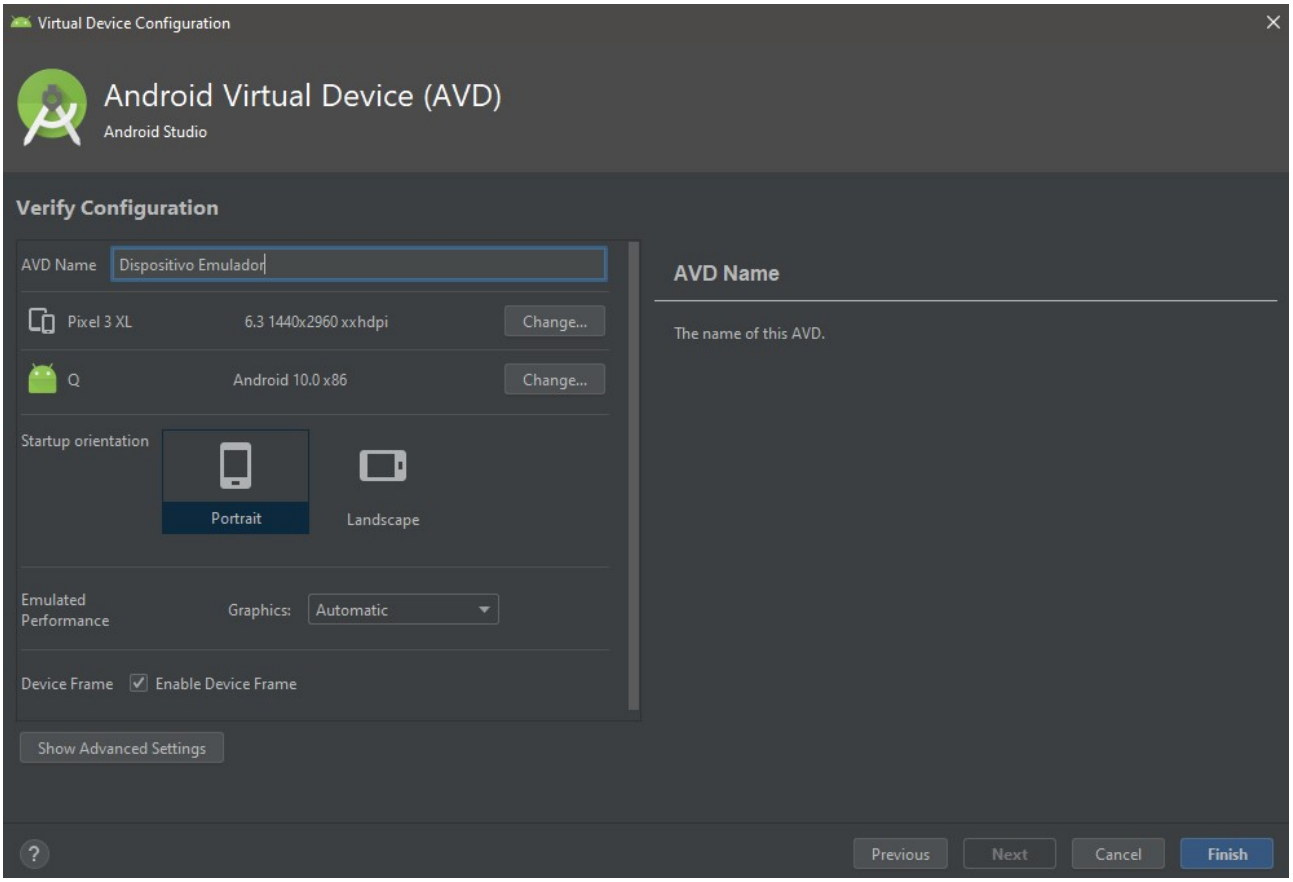


Figura B.4: Comprobamos que la configuración es la correcta

B.5. Descarga de la APK de ARCore para emuladores

Para conseguir la APK de ARCore se puede encontrar en el Github oficial de Google-Ar (web)

Figura B.5: Descargamos la APK emulator necesaria para ejecutar aplicaciones de Realidad Aumentada en el emulador del github google-ar.

B.6. Instalación de la APK ARCore para emuladores

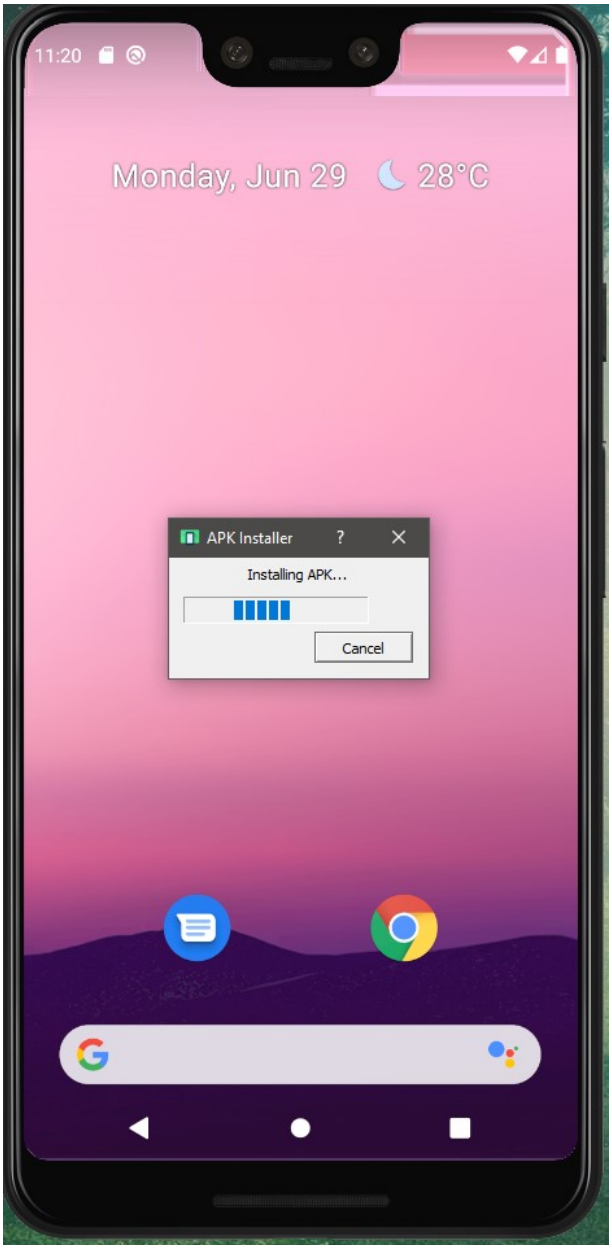


Figura B.6: Instalamos la APK descargada anteriormente arrastrando el archivo a la pantalla de la emulación.

B.7. Configuración del dispositivo

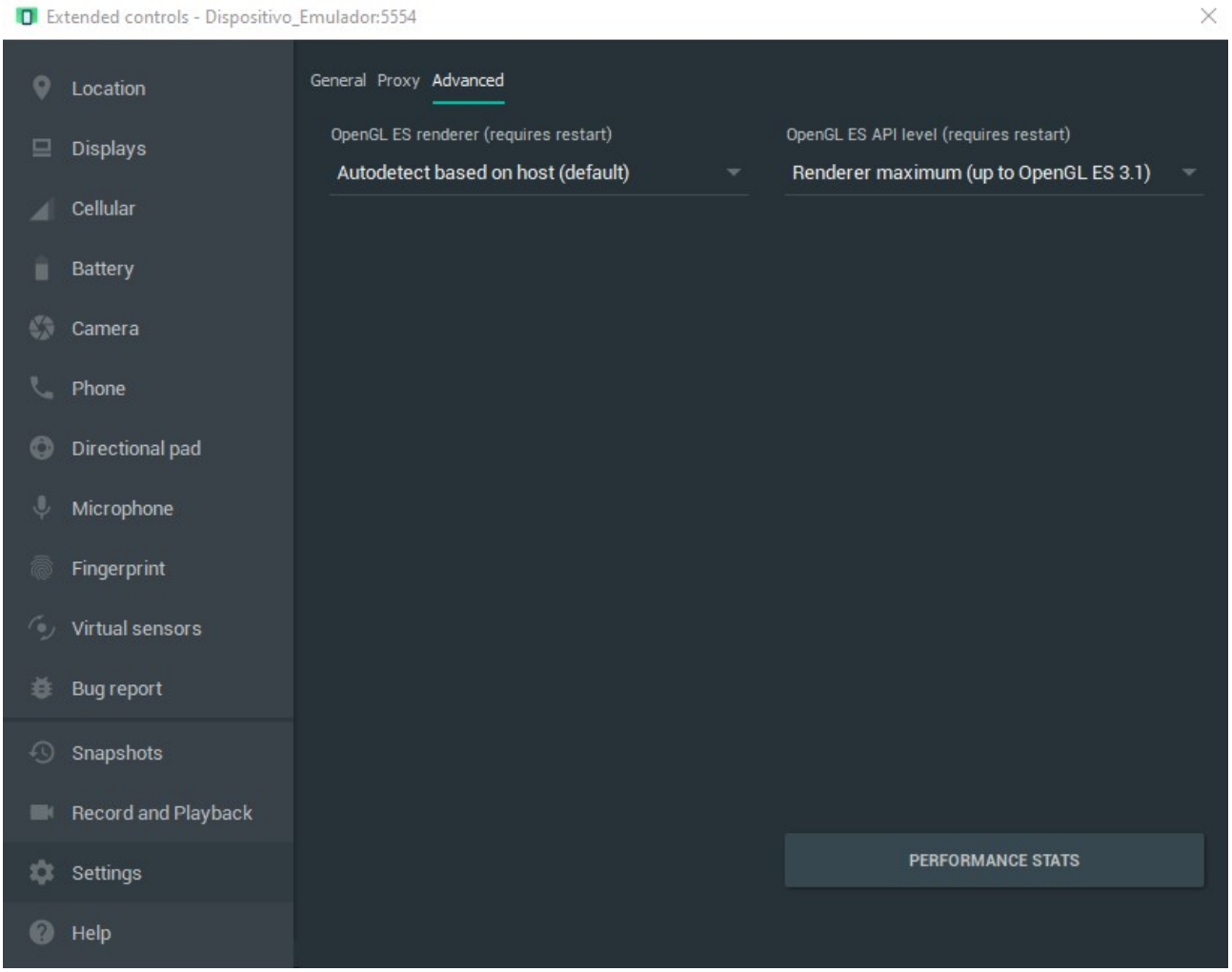


Figura B.7: Configuramos los ajustes avanzados del dispositivo forzandole a utilizar siempre una version OpenGL superior a la 3.1

B.8. Configuración de la imagen utilizada en el rastreo

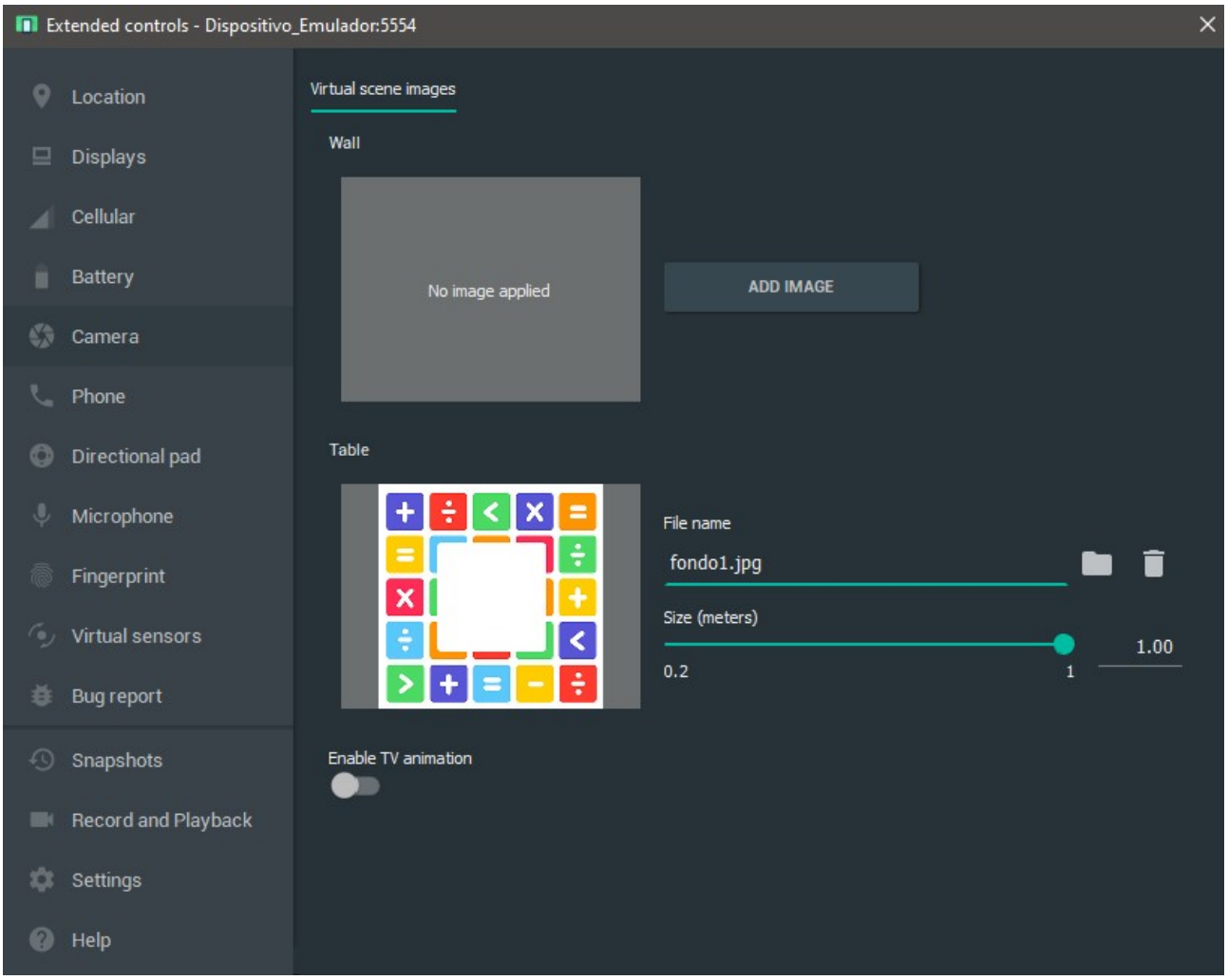


Figura B.8: En los ajustes de cámara podremos introducir las imagenes que deseamos que aparezcan en la habitación emulada.

C

Importación de assets

C.1. Creación del directorio de Objetos 3D

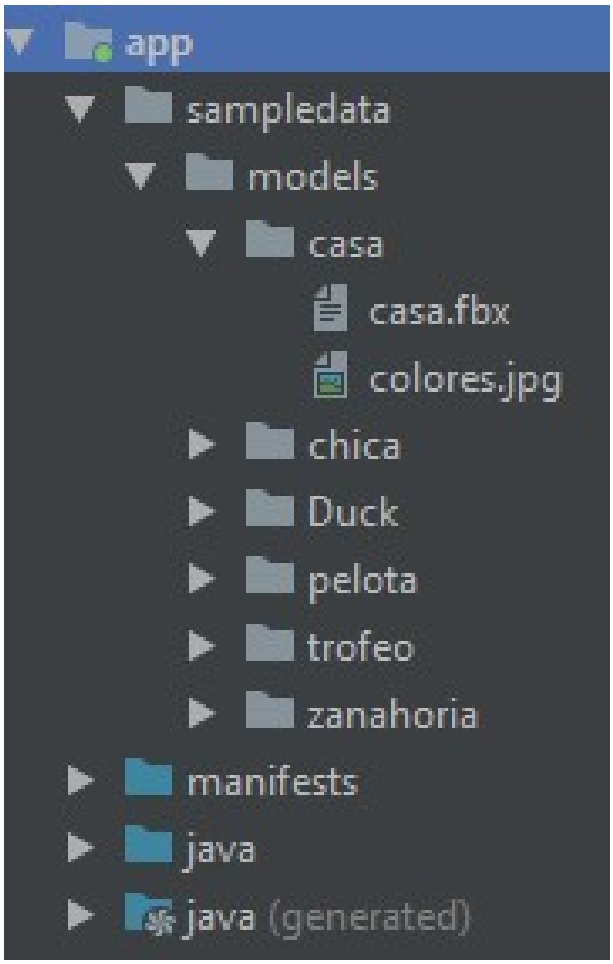


Figura C.1: Creamos el directorio sampledata en el cual se introducirán todos los modelos 3D con sus respectivas texturas que se deseen importar en la aplicación.

C.2. Importación de un Objeto 3D con Sceneform

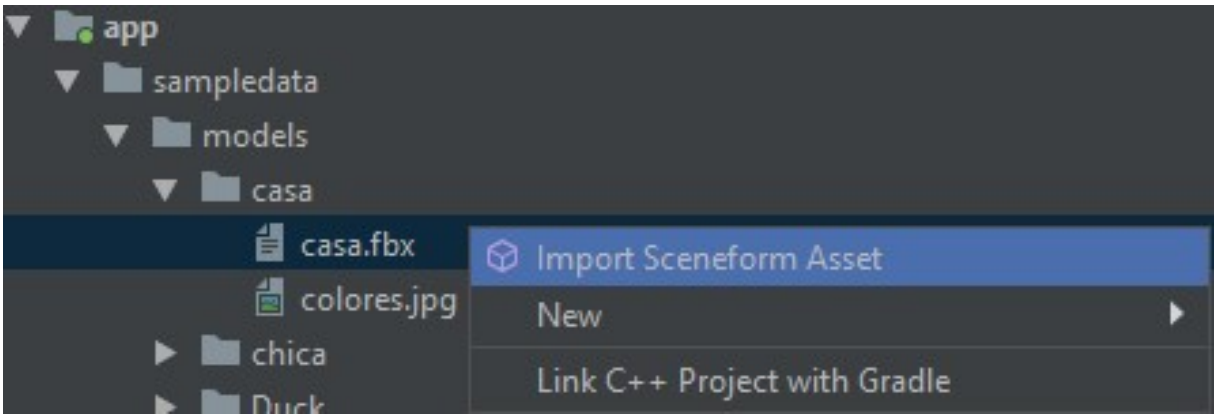


Figura C.2: Pulsamos con el botón derecho del ratón sobre el objeto que deseamos cargar en la aplicación y seleccionamos la opción Import Sceneform Asset

C.3. Configurarión del Objeto 3D

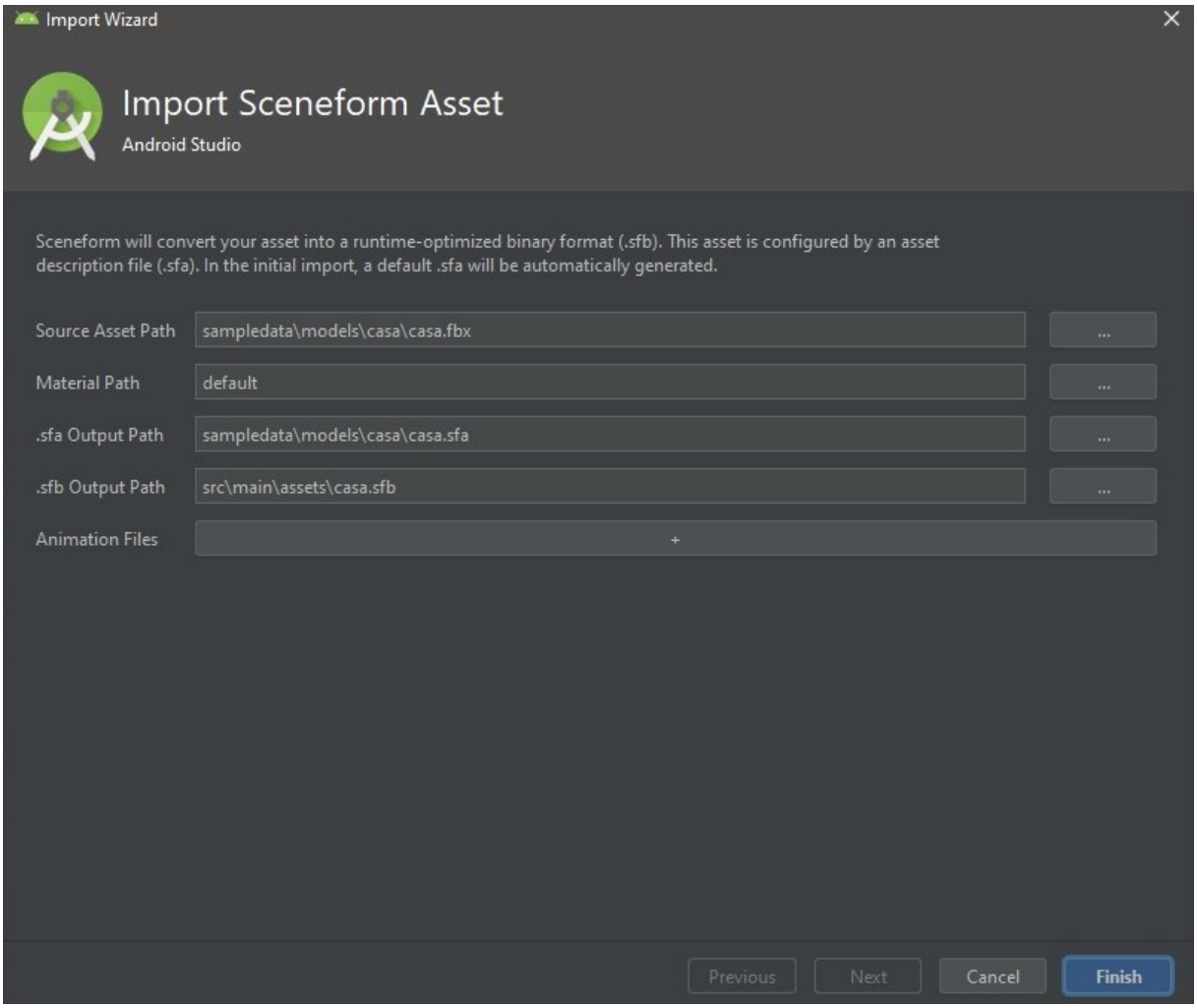


Figura C.3: Los campos mostrados se rellenarán de forma automática dando la posibilidad al programador de modificar aquellos que crea conveniente. Se da la posibilidad de adjuntar animaciones externas.

C.4. Configuración final del Objeto 3D Importado

Código C.1: En esta figura se muestran las líneas de código del modelo 3D generado en formato SFA y que permiten al desarrollador modificar ciertas propiedades del modelo importado como pueden ser su escala, rotación, materiales, etc. Esto se permite al tratarse de un objeto precompilado.

```
{
  animations: [{path: 'sampledata/models/casa/casa.fbx',},],
  materials: [
    {
      name: 'Material',
      parameters: [
        {baseColor: [1,1,1,1,],},
        {baseColorMap: 'colores',},
        {normalMap: null,},
        {interpolatedColor: null,},
        {metallic: 0,},
        {metallicMap: null,},
        {roughness: 1,},
        {roughnessMap: null,},
        {opacity: null,},
      ],
      source: 'build/sceneform_sdk/default_materials/fbx_material.sfm',
    },
  ],
  model: {
    attributes: ['Position','TexCoord','Orientation','BoneIndices','BoneWeights',],
    collision: {},
    file: 'sampledata/models/casa/casa.fbx',
    name: 'casa',
    recenter: 'root',
    scale: 0.42369400000000002,
  },
  samplers: [
    {
      file: 'sampledata/models/casa\\colores.jpg',
      name: 'colores',
      pipeline_name: 'colores.jpg',
    },
  ],
  version: '0.54:2',
}
```

C.5. Previsualización del avatar

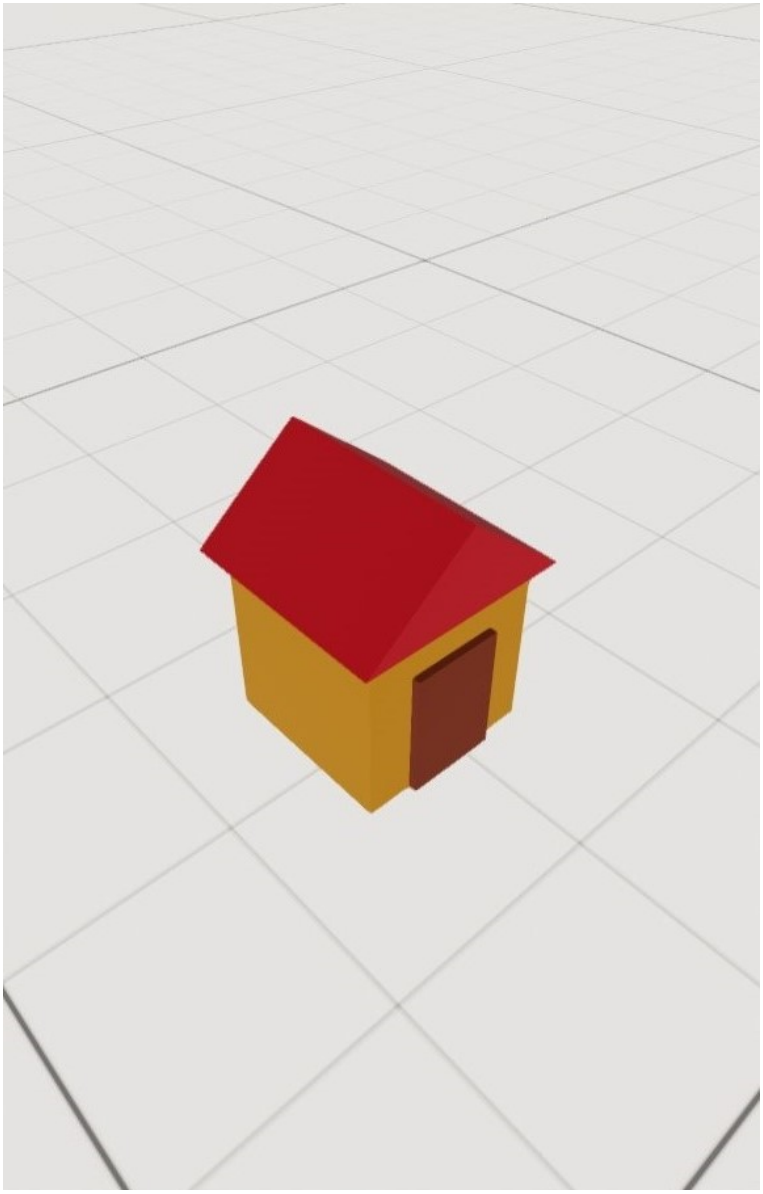


Figura C.4: Previsualización del objeto 3D importado en Android Studio.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID